Video Compressive Sensing Using Gaussian Mixture Models

Jianbo Yang, Xin Yuan, Xuejun Liao, Patrick Llull, David J. Brady, Guillermo Sapiro and Lawrence Carin

***Abstract*—A Gaussian mixture model (GMM) based algorithm is proposed for video reconstruction from temporally-compressed video measurements. The GMM is used to model spatio-temporal video patches, and the reconstruction can be efficiently computed based on analytic expressions. The GMM-based inversion method benefits from online adaptive learning and parallel computation. We demonstrate the efficacy of the proposed inversion method with videos reconstructed from simulated compressive video measurements, and from a real compressive video camera. We also use the GMM as a tool to investigate adaptive video compressive sensing, $i.e.$, adaptive rate of temporal compression.**

***Index Terms*—Compressive sensing, Gaussian mixture model, online learning, coded aperture compressive temporal imaging (CACTI), blind compressive sensing, dictionary learning, union-of-subspace model**

I. Introduction

High-speed video cameras play an important role in capturing fast motion, of interest in many applications, from science to athletics. However, measuring high-speed video often presents a challenge to camera design. So motivated, compressive sensing (CS) [1, 2] has recently been employed to capture the information in high-frame-rate video using low-frame-rate compressive measurements [3, 4, 5].

In this paper we develop a Gaussian mixture model (GMM) based inversion method for video measured with a CS camera, of the type considered in [3, 4, 5]. The inversion is probabilistic, providing a posterior distribution for the reconstructed video. Compared with other probabilistic algorithms [6, 7], the proposed method is computationally attractive, as it yields *analytic* CS inversion and is readily amenable to parallel computation. Moreover, on account of the temporal-dependence properties of video data, we proposed an online learning algorithm to update the GMM parameters as the reconstruction proceeds, so that the need of training data is minimized. In addition to considering CS video reconstruction, we address an important issue in compressive sensing, *i.e.,* measurement adaptivity. Specifically, we propose a method to adapt the *temporal* compression rate based upon the complexity of the scene.

A. Previous Work

Significant progress in video CS has been made with a single-pixel camera [8, 9, 10, 11], based on representing a video in the Fourier domain [12] or the wavelet domain [13]. More recent video CS cameras [3, 4, 14] began to incorporate temporal compression into the systems, employing the technique of coded exposure photography [15, 16, 17, 18]. In particular, [3] introduced the per-pixel programmable compressive camera (P2C2) and used an optical flow-based algorithm for reconstruction. The idea was developed further in [4], where a dictionary-based inversion algorithm is used to improve reconstruction and staggered per-pixel electronic shutters are used to facilitate implementation by today's Complementary Metal Oxide Semiconductor (CMOS) technology. Built upon the concept of [3, 4], the coded aperture compressive temporal imaging (CACTI) camera introduced in [5, 19] features a mechanically-translated coding element, which modulates high-speed motion at low power and cost.

The focus of this paper is on new video reconstruction algorithms. A number of inversion algorithms have been developed for video CS in the past few years. Wakin et al. [13] used a 3D wavelet based inversion algorithm to achieve video CS for single-pixel cameras. A more sophisticated method was developed in [20], where the evolution of a scene is modeled by a linear dynamical system (LDS) and the LDS's parameters are estimated from the compressive measurements. Park and Wakin [21, 22] developed a coarse-to-fine algorithm which alternates between temporal motion estimation and spatial frame reconstruction in the wavelet-domain. In [23] an algorithm for estimating optical flow between images was described, and it was used for video reconstruction in [3]. Besides [3], Mun and Fowler [24] used optical flow to estimate the motion field of the video frames, and the estimation is performed alongside reconstruction of the video frames.

Other popular algorithms for video CS have been based on total variation (TV) [14, 25] and dictionary learning [4, 14]. TV methods assume that the gradient of each video frame is sparse and attempts to minimize the ℓ_p norm of the gradient frames summed over all time steps. Dictionary-based methods represent each video patch as a sparse linear expansion in the dictionary elements. The dictionary is often learned offline from training video and the sparse coefficients of a video patch can be achieved by sparse-coding algorithms. The video inversion algorithm proposed in this paper is based on a Gaussian mixture model of spatiotemporal patches, and its connection to dictionary-based algorithms is presented in Section III-C.

B. Gaussian Mixture Model

The GMM is used widely in various learning tasks, including classification and segmentation [26, 27], as well as image denoising, inpainting and deblurring [28, 29]. Recent work has

The authors are with Electrical & Computer Engineering Department, Duke University, Durham, NC 27708-0291.

demonstrated that the GMM is also a powerful tool for image reconstruction in still-image CS [28, 30].

In this paper we extend the GMM to compressive video. Specifically, when performing CS inversion, the video's pixel volume is divided into a set of space-time patches. All pixels within a patch are assumed drawn (at once) from a GMM. The covariance matrix of each mixture component defines a linear subspace (like in principal component analysis), and therefore the GMM representation is closely related to the union-of-subspace model [31, 32, 33]. Within the GMM construction, each patch of data is effectively assumed drawn from one of the subspaces, from among the union of subspaces. The posterior distribution for the underlying data associated with the compressive measurement of the patch is shown here to also be an (updated) GMM; thus, the estimated video is a weighted average across multiple of the subspaces in the union. The connection of the GMM to a union of subspaces links the proposed approach to dictionary-learning-based CS inversion [4]. The advantage of using the GMM over dictionary-learning-based approaches is, as demonstrated below, that the CS inversion is analytic (the posterior distribution on the underlying patch pixels is represented by an updated GMM, with analytic parameter update).

In previous dictionary learning [4] and GMM-based CS inversion [30], it was assumed that the dictionary/GMM was learned offline. This assumes that one has access to an appropriate set of training data. To address this limitation, we propose an online-updated GMM algorithm, that adapts the GMM parameters as the reconstruction proceeds. We demonstrate that online updates of the underlying GMM mitigates the need for training data, as after sufficient data are observed the data under test drives the GMM, not prior training data.

C. Measurement Adaptivity

Studies have shown that improved CS performance can be achieved when projection/measurement matrices are designed to adapt to the underlying signal of interest [34, 35]. All of this previous work has been developed for CS measurement of a *single* image. After exhaustive study of this matter, we found that within the constraints imposed by the form of the compressive video cameras discussed above ($e.g.$, positive codes), there is only modest gain accrued in trying to optimize the coding mask, relative to just employing random design. Variations of the methods in [34, 35] were examined in detail, and this was determined to not be a fruitful research direction (for video reconstruction under realistic hardware design constraints).

However, a second opportunity for adaptivity concerns the details of the designed video CS cameras considered in [3, 4, 5]. In each of these cameras, the high-speed video frames are coded with a mask, and then the coded frames over time window Δ_t are summed to constitute a single measurement. The appropriate value for Δ_t may vary depending on what is happening in the scene, and therefore there is an opportunity for adaptivity. We consider such adaptivity in the context of the framework proposed here.

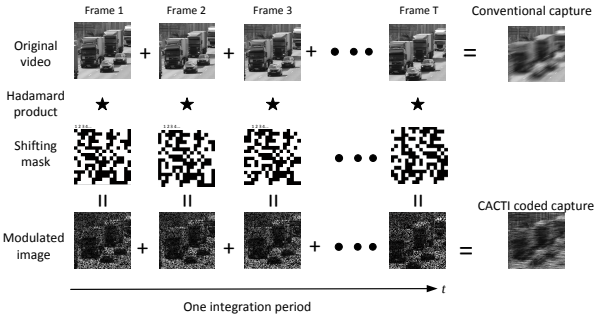


Fig. 1. Illustration of the coding mechanisms within the CACTI hardware system. The first row depicts T original video frames; the second row denotes the mask with which each frame is multiplied (black is zero, white is one). In CACTI, the same code is shifted to constitute a series of frame-dependent codes. Note that the positions of columns 1,2,3,4... of the mask are translated in one integration period. Finally, the T frames are summed to constitute one measurement, as shown at the right-bottom.

D. Paper Organization

The remainder of the paper is organized as follows. Section II introduces the CACTI compressive video camera that we will use to motivate our simulated experiments, and it will also be a source of real data. GMM-based CS inversion for video is developed in Section III. Section IV considers adapting Δ_t based on the inferred complexity of the scene under test. An extensive set of experimental results are presented in Section V, and Section VI provides conclusions and suggestions for future work.

II. Coded Aperture Compressive Temporal Imaging (CACTI)

The CACTI measurement framework is summarized in Fig. 1, and it is this system to which all experiments below are applied. The proposed modeling and inversion framework may also be applied to data collected by related compressive video cameras [3, 4]. The distinction between CACTI and the cameras in [3, 4] is manifested in how the coding process (middle row of Figure 1) is implemented. In [3, 4] a Liquid-Crystal-on-Silicon (LCoS) device is employed to achieve per-pixel modulation, which affords greater coding flexibility than the proposed approach, but at greater implementation complexity. In CACTI, a *single* binary code is employed; this code is mechanically shifted via a pizeoelectronic translator. The single binary code to be shifted is designed by randomly drawing 1/0 values at every pixel value, with a 0.5 probability of a 1. The 0.5 probability was found to give best performance. In *numerical* experiments, we considered the case for which each of the codes on the second row of Figure 1 were drawn uniformly at random (*not* translating a single mask), to examine the performance degradation manifested by requiring the translation of the same shifted code (as actually implemented in CACTI). In those experiments, it was found that the CACTI design yielded almost as good CS recovery performance as the totally random design (each frame coded with a uniquely drawn random code). Therefore, the CACTI design manifests significant ease and simplicity of implementation, with minimal degradation in performance.

To explain the measurement process mathematically, consider a two-dimensional datacube, with one spatial dimension, s, and time t. In the actual camera the datacube is three-dimensional, with two spatial dimensions. Let $x(s,t)$ represent the *continuous* space-time datacube that we wish to measure. The function $\mathcal{T}(s)$ represents the fixed binary code, which is a function of space dimension s; the spatial translation of the code is characterized by the function $\tilde{s}(t)$. The discrete measurement at space-time point $(\bar{s},\bar{t})$ is represented as

$$g(\bar{s},\bar{t}) = \iint x(s,t)\mathcal{T}[s-\tilde{s}(t)]\text{rect}\left(\frac{s-\bar{s}}{\Delta_s}\right)\text{rect}\left(\frac{t-\bar{t}}{\Delta_t}\right)dsdt,$$

where $\text{rect}[(s-\bar{s})/\Delta_s]$ is a rectangular window function that has unit amplitude over spatial support Δ_s centered at $\bar{s}$, and it is zero elsewhere (and similarly for the temporal rect function). In this setup the spatial coding rate on the mask $\mathcal{T}(s)$ could be different from that of the sampling system, denoted by Δ_s. In the actual CACTI camera we consider below, the size of the pixels in the code $\mathcal{T}(s)$ are the same as Δ_s, and CS decompression is performed on the scale of Δ_s. This implies that there is no spatial compression in the system, with compression manifested in time, as detailed next.

The goal is to perform approximate recovery of $x(s,t)$ for $t \in [\bar{t}-\Delta_t/2, \bar{t}+\Delta_t/2]$, based upon measuring $g(\bar{s},\bar{t})$. The recovered $x(s,t)$ is discretized in space and time. The spatial sampling rate is Δ_s, with the discrete spatial samples centered at all spatial measurement points $\bar{s}$. Concerning the temporal sampling rate of the recovered data cube, consider $x(s,n\delta_t)$ for $\delta_t = \Delta_t/T$, and $n = 1,\ldots,T$; these correspond to the underlying datacube at T discrete points in time. When we convert the inversion problem into a linear matrix, the question concerns choice of T, which defines the number of frames to be estimated from a single compressive measurement that integrates over time Δ_t.

Recall that the mask/code in CACTI is moving *continuously* as a function of time, as dictated by $\tilde{s}(t)$. Therefore, we must choose T large enough such that the mask is *approximately* fixed in space over time period δ_t (to justify our conversion to a matrix equation, as discussed next). On the other hand, if T is made too large, the number of unknowns to be recovered may become excessive. To clarify these points, we next express the inversion problem explicitly in matrix form; for that we move to the actual three-dimensional datacube of interest.

Let $\mathbf{X} \in \mathbb{R}^{N_x \times N_y \times T}$ represent the underlying *discretized* datacube to be recovered; the spatial sampling rate, that defines N_x and N_y, is specified by the hardware sampling rate Δ_s. As discussed above, T is a parameter that may be set, which impacts the accuracy of the approximate linear-equation representation of the system. Toward that linear equation, let $\mathbf{Y} \in \mathbb{R}^{N_x \times N_y}$ represent the two-dimensional discrete CACTI measurement, corresponding to one measurement window Δ_t. The relationship between the pixels of $\mathbf{X}$ and those of $\mathbf{Y}$ is specified as

$$\mathbf{Y}_{i,j} = [\mathcal{A}_{i,j,1}\ \mathcal{A}_{i,j,2}\ \cdots\ \mathcal{A}_{i,j,T}]\,[\mathbf{X}_{i,j,1}, \mathbf{X}_{i,j,2}, \ldots, \mathbf{X}_{i,j,T}]' \quad (1)$$

where $'$ denotes the transpose of a vector, $\mathbf{Y}_{i,j}$ is component (i,j) of $\mathbf{Y}$ and $\mathbf{X}_{i,j,n}$ is associated with the same spatial pixels (i,j), and temporal bin n. The components $\mathcal{A}_{i,j,n} \in \{0,1\}$, as dictated by the binary mask, and it is assumed that the mask is approximately fixed in space over time period $\delta_t = \Delta_t/T$, as discussed above. Therefore, within the approximations specified, each pixel in $\mathbf{Y}$ constitutes a weighted sum of the corresponding pixels in the T frames of $\mathbf{X}$, where the weights are binary.

Anticipating the inversion algorithm to be considered in the next section, we partition $\mathbf{Y}$ into a set of patches, where each square patch is composed of P^2 contiguous pixels. We may consider all possible overlapping patches, and let $\mathbf{y}_m$ represent the m^{th} patch. For each m, we may write $\mathbf{y}_m = \mathbf{\Phi}_m\mathbf{x}_m$ where $\mathbf{y}_m \in \mathbb{R}^{P^2}$, $\mathbf{x}_m \in \mathbb{R}^{P^2T}$, and $\mathbf{\Phi}_m \in \{0,1\}^{P^2 \times P^2T}$, where the pixels in two-dimensional ($\mathbf{y}_m$) and three-dimensional ($\mathbf{x}_m$) settings have been "vectorized."

For each m, the objective is to recover $\mathbf{x}_m$ based upon the measured $\mathbf{y}_m$. Since the dimension of the item to be inferred is much larger than that of the associated measurement, this inversion is ill-posed unless constraints or prior knowledge is imposed with respect to $\mathbf{x}_m$. This is done in the context of a Gaussian mixture model (GMM).

III. Gaussian Mixture Model for Video Reconstruction

A. GMM-Based Inversion

Consider a total of M two-dimensional measurement patches $\{\mathbf{y}_m\}_{m=1,M}$, each of which has an associated three-dimensional space-time signal $\mathbf{x}_m$ that we wish to recover. Each of the inversions are performed independently, and therefore there is significant opportunity for parallel acceleration. The final estimate of any pixel in $\mathbf{X}$ is its average from all recovered patches in which it resides. Each measurement may be expressed as

$$\mathbf{y}_m = \mathbf{\Phi}_m\mathbf{x}_m + \boldsymbol{\epsilon}_m, \quad \forall\ m = 1,\cdots,M, \quad (2)$$

where we now consider measurement noise, and/or model mismatch error $\boldsymbol{\epsilon}_m \in \mathbb{R}^{P^2}$.

In the analysis that follows, we assume that the GMM representation for $\mathbf{x}_m$ corresponds to a P^2T-dimensional signal, where we recall that T temporal frames are recovered for each CS measurements. We may alternatively assume $\mathbf{x}_m \in \mathbb{R}^{P^2T_0}$, where $T_0 = nT$, for $n > 1$. In this case multiple temporal CS measurements are employed to recover the underlying $\mathbf{x}_m$.

We assume $\mathbf{x}_m$ is drawn from a GMM, as

$$\mathbf{x}_m \sim \sum_{k=1}^{K} \lambda_k \mathcal{N}(\mathbf{x}_m|\boldsymbol{\mu}_k, \mathbf{\Sigma}_k), \quad (3)$$

where $\boldsymbol{\mu}_k$, $\mathbf{\Sigma}_k$, and λ_k are the mean, covariance matrix, and weight of the k^{th} Gaussian component, respectively ($\lambda_k > 0$ and $\sum_{k=1}^{K}\lambda_k = 1$). These parameters can be estimated by the expectation-maximization (EM) [36] algorithm, based on training video patches. One may also employ more-sophisticated approaches, such as nonparametric Bayesian models [30, 37], to estimate the GMM parameters, from which an appropriate number of mixture components K can be inferred. Here we simply set K, and the EM algorithm

is employed. The GMM in (3) constitutes our prior for the underlying $\mathbf{x}_m$.

Concerning the measurement model, it is assumed that $\boldsymbol{\epsilon}_m \sim \mathcal{N}(\boldsymbol{\epsilon}_m|0, \boldsymbol{R})$, where $\boldsymbol{R}$ is a $P^2 \times P^2$ covariance matrix, and therefore

$$\mathbf{y}_m|\mathbf{x}_m \sim \mathcal{N}(\mathbf{y}_m|\boldsymbol{\Phi}_m\mathbf{x}_m, \boldsymbol{R}), \tag{4}$$

Applying Bayes' rule results in the posterior

$$p(\mathbf{x}_m|\mathbf{y}_m) = \sum_{k=1}^{K} \tilde{\lambda}_{mk}\mathcal{N}(\mathbf{x}_m|\tilde{\boldsymbol{\mu}}_{mk}, \tilde{\boldsymbol{\Sigma}}_{mk}), \tag{5}$$

with

$$\tilde{\lambda}_{mk} = \frac{\lambda_k\mathcal{N}(\mathbf{y}_m|\boldsymbol{\Phi}_m\boldsymbol{\mu}_k, \boldsymbol{R} + \boldsymbol{\Phi}_m\boldsymbol{\Sigma}_k\boldsymbol{\Phi}_m^T)}{\sum_{l=1}^{K}\lambda_l\mathcal{N}(\mathbf{y}_m|\boldsymbol{\Phi}_m\boldsymbol{\mu}_l, \boldsymbol{R} + \boldsymbol{\Phi}_m\boldsymbol{\Sigma}_l\boldsymbol{\Phi}_m^T)}, \tag{6}$$

$$\tilde{\boldsymbol{\Sigma}}_{mk} = (\boldsymbol{\Phi}_m^T\boldsymbol{R}^{-1}\boldsymbol{\Phi}_m + \boldsymbol{\Sigma}_k^{-1})^{-1},$$

$$\tilde{\boldsymbol{\mu}}_{mk} = \tilde{\boldsymbol{\Sigma}}_{mk}(\boldsymbol{\Phi}_m^T\boldsymbol{R}^{-1}\mathbf{y}_m + \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k), \tag{7}$$

which is also a GMM [30]. The analytic inversion expression (5) results in efficient reconstruction of every patch.

The posterior distribution on $\mathbf{x}_m$ is of interest for characterizing confidence in the inversion. We often are interested in reporting a single estimate of the underlying signal, and for this we may employ the conditional expectation:

$$\mathbb{E}(\mathbf{x}_m|\mathbf{y}_m) = \int \mathbf{x}_m p(\mathbf{x}_m|\mathbf{y}_m)\mathrm{d}\mathbf{x}_m = \sum_{k=1}^{K}\tilde{\lambda}_{mk}\tilde{\boldsymbol{\mu}}_{mk} \stackrel{\text{def}}{=} \tilde{\mathbf{x}}_m. \tag{8}$$

We term this implementation of the above framework *offline* GMM-based inversion, as all patches use the same offline-learned GMM parameters for the prior (which assumes appropriate training videos).

B. Online-Updated GMM for Video Reconstruction

Though the GMM-based inversion based on an offline-learned GMM effectively reconstructs CS-measured video, this implementation does not explicitly exploit temporal dependence among patches, and it assumes appropriate training data $\boldsymbol{D}_0$ (analogous to previous dictionary-learning based inversion, in which the dictionary was learned offline [4]).

The inversion algorithm recovers video in patches, and the temporal length of each patch is T_0, recalling that T_0 is a multiple of T. Let $\boldsymbol{D}_0$ represent the patches in the training data, and $\boldsymbol{D}_l$ represents the *recovered* patches with support within time window $(l-1)T_0 < t \leq lT_0$, for $l = 1, \ldots, L-1$. Based on $\{\boldsymbol{D}_0, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_{L-1}\}$, we wish to update the GMM prior, for use when performing inversion of CS-measured patches within the time window $(L-1)T_0 < t \leq LT_0$. Clearly the amount of data grows with increasing L, and therefore we wish to prune the data used for GMM learning, with an emphasis on the most recent data.

Assume that M_L represents the total number of video patches we wish to train the GMM with. We may define $\pi_l \in (0, 1)$ with $\sum_{l=0}^{L-1}\pi_l = 1$, and sample $[\pi_l M_L]$ patches from $\boldsymbol{D}_l$, uniformly at random, where $[\cdot]$ represents the nearest integer. By making $\pi_l < \pi_{l+1}$, the model emphasizes more recent data over later data, and eventually the original training data $\boldsymbol{D}_0$ becomes irrelevant.

The parameters $\{\pi_l\}$ are here defined by setting $\xi_l = (l+1)^\kappa$ and $\pi_l = \xi_l / \sum_{l'=0}^{L-1}\xi_{l'}$, where $\kappa \in [0, \infty]$ controls the decay rate of influence on the adapted GMM. A similar strategy has been used in online latent Dirichlet allocation (LDA) [38] and online beta process factor analyzer (BPFA) [39].

There are many ways in which such adaptivity may be performed, and by which the sets $\{\boldsymbol{D}_0, \boldsymbol{D}_1, \ldots, \boldsymbol{D}_{L-1}\}$ may be defined. For example, a given batch of data $\boldsymbol{D}_l$ may consider patches within time windows that are larger than T_0. Further, rather than adapting the GMM continuously, which may be expensive, the GMM may be updated after periodically specified intervals. As validated in the numerical experiments, the online-updated GMM method can mitigate the need for training data, as after sufficient data are observed, the data under test drives the GMM, not the prior training data.

The sketch of the proposed GMM-based CS reconstruction algorithm is shown in Algorithm 1.

Algorithm 1 The GMM-based Reconstruction Algorithm

Input: Measurements $\mathbf{Y}$, projection matrix $\boldsymbol{\Phi}$ and integration window T.

Output: The reconstructed video $\mathbf{X}$.

1. Initialize the GMM parameters by running EM algorithm based on training data D_0. Let $T_0 = nT$, and n is an integer set by user.
2. Let $l = 1$.

repeat

3. Reconstruct video patches in $\boldsymbol{D}_l$ with support within time window $(l-1)T_0 < t < lT_0$, based on the inversion formula (5).
4. Update the GMM parameters according to the procedure in Section III.B (only for the online-updated GMM method).
5. Convert video patches in $\boldsymbol{D}_l$ into a video clip and put this video clip into the corresponding position in $\mathbf{X}$.

until All measurements are used, *i.e.*, $l = L$.

C. Connections to Existing CS Inversion Methods

1) Dictionary Learning: Dictionary learning has achieved success in image CS reconstruction [6, 40, 41] and has been recently applied to invert compressive video measurements [4]. In these methods, an overcomplete dictionary $\boldsymbol{W} \in \mathbb{R}^{d_x \times I}$ ($I > d_x$) is considered for representation of the signal $\mathbf{x} \in \mathbb{R}^{d_x}$, with a sparse vector of dictionary coefficients $\boldsymbol{f} \in \mathbb{R}^I$, *i.e.*, $\mathbf{x} = \boldsymbol{W}\boldsymbol{f} + \boldsymbol{\epsilon}$. The dictionary $\boldsymbol{W}$ is *dataset-specific*, and its superiority over universal bases such as the discrete cosine transform (DCT) and the wavelet transform has been demonstrated in various image and video applications [4, 6, 28, 40, 41]. State-of-the-art dictionary learning methods include K-SVD [40], method of optimal directions (MOD) [42] and beta process factor analysis (BPFA) [6], among others. Extending these dictionary learning approaches, block-structured dictionary learning (BSDL) has recently been considered. In BSDL, the signal is assumed to be drawn

from a union of a small number of disjoint subspaces. The recent BSDL methods, such as Block BP [31], Block OMP [32], group LASSO [43], block sparse dictionary optimization (BSDO) [44] have yielded favorable results over standard dictionary learning methods. It is worth noting that most existing dictionary learning methods have high computational cost, in particular when compared with the GMM learning and reconstruction procedure. This computational problem is exacerbated in BSDL, as the block structure of dictionaries needs to be determined additionally [44].

We now illustrate that the GMM-based inversion method is related to dictionary learning, BSDL in particular. Recall (8) that the reconstruction $\tilde{\mathbf{x}}$ yielded by the GMM-based inversion method is expressed by $\tilde{\mathbf{x}} = \sum_{k=1}^{K} \tilde{\lambda}_k \tilde{\boldsymbol{\mu}}_k$. For convenience, we rewrite the mode $\tilde{\boldsymbol{\mu}}_k$ as

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_k &= \tilde{\boldsymbol{\Sigma}}_k(\boldsymbol{\Phi}^T\boldsymbol{R}^{-1}\mathbf{y} + \boldsymbol{\Sigma}_k^{-1}\boldsymbol{\mu}_k) && (9)\\
&= \mathbf{E}_k \underbrace{\boldsymbol{\Lambda}_k^{\frac{1}{2}}[\boldsymbol{I} + \boldsymbol{\Lambda}_k^{\frac{1}{2}}(\boldsymbol{\Phi}\mathbf{E}_k)^T\boldsymbol{R}^{-1}\boldsymbol{\Phi}\mathbf{E}_k\boldsymbol{\Lambda}_k^{\frac{1}{2}}]^{-1}\boldsymbol{\Lambda}_k^{\frac{1}{2}}(\boldsymbol{\Phi}\mathbf{E}_k)^T\boldsymbol{R}^{-1}\mathbf{y}}_{\stackrel{\text{def}}{=}\boldsymbol{f}_k}\\
&\quad + \underbrace{\mathbf{E}_k\boldsymbol{\Lambda}_k^{\frac{1}{2}}[\boldsymbol{I} + \boldsymbol{\Lambda}_k^{\frac{1}{2}}(\boldsymbol{\Phi}\mathbf{E}_k)^T\boldsymbol{R}^{-1}\boldsymbol{\Phi}\mathbf{E}_k\boldsymbol{\Lambda}_k^{\frac{1}{2}}]^{-1}\boldsymbol{\Lambda}_k^{-\frac{1}{2}}\mathbf{E}_k^T\boldsymbol{\mu}_k}_{\stackrel{\text{def}}{=}\boldsymbol{\eta}_k} && (10)\\
&= \mathbf{E}_k\boldsymbol{f}_k + \boldsymbol{\eta}_k, && (11)
\end{aligned}$$

in which the matrix inversion lemma and eigenvalue decomposition $\boldsymbol{\Sigma}_k = \mathbf{E}_k\boldsymbol{\Lambda}_k\mathbf{E}_k^T$ are used.

When rewriting (11) as $\tilde{\boldsymbol{\mu}}_k - \boldsymbol{\eta}_k = \mathbf{E}_k\boldsymbol{f}_k$, the residual of $\tilde{\boldsymbol{\mu}}_k$ and $\boldsymbol{\eta}_k$ can be represented by a sparse representation $\boldsymbol{f}_k$ on the base $\mathbf{E}_k$. Herein, $\mathbf{E}_k$ contains eigenvectors of covariance matrix of the k^{th} Gaussian component. According to (10), the representation$\boldsymbol{f}_k$ can also be written as $\boldsymbol{f}_k = [\boldsymbol{\Lambda}_k^{-1} + (\boldsymbol{\Phi}\mathbf{E}_k)^T\boldsymbol{R}^{-1}\boldsymbol{\Phi}\mathbf{E}_k]^{-1}(\boldsymbol{\Phi}\mathbf{E}_k)^T\boldsymbol{R}^{-1}\mathbf{y}$, and this implies that $\boldsymbol{f}_k$ is estimated by the *weighted* ℓ_2-norm regularized regression,

$$\boldsymbol{f}_k = \arg\min_{\boldsymbol{f}} \left(\|\mathbf{y} - \boldsymbol{\Phi}\mathbf{E}_k\boldsymbol{f}\|_2^2 + \sigma^2\boldsymbol{f}'\boldsymbol{\Lambda}_k^{-1}\boldsymbol{f}\right), \qquad (12)$$

assuming $\boldsymbol{R} = \sigma^2\boldsymbol{I}, \sigma \in \mathbb{R}$. Since eigenvalues contained in $\boldsymbol{\Lambda}_k$ are dominated by a few elements [45], the solution $\boldsymbol{f}_k$ in (12) is a sparse vector. It is easy to verify the sparsity of $\boldsymbol{f}_k$ in (10).

Based on the above analysis, the proposed GMM-based inversion method resembles dictionary learning. After applying the derivation in (9) - (11) for all $\tilde{\boldsymbol{\mu}}_k, k = 1, \ldots, K$, the dictionary $\boldsymbol{W}$ can be set as a union of K bases, *i.e.*, $\boldsymbol{W} = \mathbf{E}_1 \cup \cdots \cup \mathbf{E}_K$. However, rather than treat K bases $\mathbf{E}_1 \ldots \mathbf{E}_K$ equally like BSDL, GMM-based inversion uses the probabilistic weights $\lambda_1, \ldots, \lambda_K$ to specify the importance of the K bases. These weights are involved in the inversion procedure as indicated by $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_K$ in (5). Note that $\lambda_1, \ldots, \lambda_K$ are generally not sparse, while $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_K$ are often sparse. This implies that a patch to be reconstructed can be drawn from a probabilistic union of a few of K subspaces; each subspace is spanned by the leading eigenvectors of the corresponding component's covariance matrix. The GMM-based inversion method simultaneously fulfills the following three tasks: dictionary learning, determination of block structure, and sparse signal representation. The associated computation is very fast due to the analytic inversion in (5).

2) Piecewise Linear Estimator: Piecewise Linear Estimators (PLE) [28, 46] constitute another recent GMM-based inversion method used for still-image compressive sensing. By dividing the image signal into a collection of $P \times P$ patches, PLE also models the image patch to be reconstructed by a GMM. Given the compressed measurement of each patch, the image patch can be reconstructed based on a Wiener filter, that also enjoys an analytic form.

Though the basic idea of PLE is similar to that of the proposed method, there are the following important differences between PLE and the proposed method: (i) PLE constrains that the reconstructed patch is drawn from one of the K Gaussian distributions. Equivalently, PLE assumes one-block sparsity in BSDL. This hard assignment is different from our soft assignment as shown in (8). In the experiments of video CS, we observe that more than one Gaussian components often contribute to reconstructing a video patch. (ii) PLE employs a different initialization procedure, by exploiting the particular properties of images. Specifically, as claimed in [28, 46], a 2D image patch often has certain patterns, such as dominant spatial directions. Thus, in the initialization of PLE, $K-1$ Gaussian components correspond to $K-1$ angles that are uniformly sampled from 0 to π. The PCA spaces of these $K-1$ components are learned based on the synthetic directional image patches. The K^{th} Gaussian component employing the DCT as the basis captures the isotropic image patterns. This initialization strategy is not straightforward to extend to the video case, as it is not clear how to define the dominant direction for a 3D video patch, especially when T_0 is large. PLE is also similar to BSDL in the sense that all K Gaussian components are treated equally. If one writes the GMM prior of PLE in the form (3), it leads to $\lambda_1 = \lambda_2 = \cdots = \lambda_K$, which is not the case for the proposed GMM-based inversion method. (iii) In PLE, the GMM's K components are separately tackled in the procedures of initialization, parameters adaptation, and inversion. By contrast, in the proposed GMM-based inversion method, the K Gaussian components are always jointly involved in these three procedures.

D. Computational Complexity

We analyze the computational complexity of the proposed GMM-based inversion method. As the *online* GMM method presented in Section III-B iterates over different time windows, we focus on the computational complexity within one of them. Also, we describe the computational complexity in two different scenarios: standard computation and accelerated computation.

1) Standard Computation: For convenience, we rewrite parameters of (5) as below:

$$\tilde{\boldsymbol{\Sigma}}_k = \boldsymbol{\Sigma}_k - \boldsymbol{\Sigma}_k\boldsymbol{\Phi}^T \underbrace{(\boldsymbol{R} + \boldsymbol{\Phi}\boldsymbol{\Sigma}_k\boldsymbol{\Phi}^T)^{-1}}_{\stackrel{\text{def}}{=}\mathbf{A}} \boldsymbol{\Phi}\boldsymbol{\Sigma}_k; \qquad (13)$$

$$\tilde{\boldsymbol{\mu}}_k = \boldsymbol{\Sigma}_k\boldsymbol{\Phi}^T \underbrace{(\boldsymbol{R} + \boldsymbol{\Phi}\boldsymbol{\Sigma}_k\boldsymbol{\Phi}^T)^{-1}}_{=\mathbf{A}}(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}_k) + \boldsymbol{\mu}_k. \qquad (14)$$

The proposed GMM-based inversion method includes two parts: EM model learning and patch-based inversion. The EM

TABLE I
THE COMPUTATIONAL COSTS OF THE PROPOSED GMM-BASED INVERSION METHOD FOR CASES OF STANDARD AND ACCELERATED COMPUTATION. K IS THE NUMBER OF GMM COMPONENTS, M_L IS THE NUMBER OF SAMPLES USED IN EM TRAINING IN THE L^{th} TIME WINDOW.

Scenario 1: Standard Computation	
Patch-independent (for all patches in an integration window)	
EM algorithm	$O(M_L K d_x^2)$ [47])
Patch-dependent (for one patch)	
Compute $\mathbf{A}$	$O(K d_y^3/3)$ [28]
Compute (13) and (14) without $\mathbf{A}$	$O(K d_y d_x)$
Compute (6) without $\mathbf{A}$	$O(K d_y^2)$
Scenario 2: Accelerated Computation	
Patch-independent (for all patches in an integration window)	
EM algorithm	$O(M_L K d_x^2)$)
Eigendecomposition	$O(K d_x^3)$
Patch-dependent (for one patch)	
Compute $\boldsymbol{G}$	$O(K h_k^3/3)$
Compute (15) and (16) without $\boldsymbol{G}$	$O(K h_k d_x)$
Compute (6) without $\boldsymbol{G}$	$O(K h_k d_y)$

algorithm is called once for every T_0 frames reconstruction. Particularly, in *offline* learning, it is only called once for the entire video reconstruction. For the part of patch-based inversion, the analytic form exists as shown in (5), and the main cost lies in matrix inversion for computing $\mathbf{A}$ in (6), (13) and (14), whose computational cost is $\frac{d_x^3}{3} + d_x^2 \approx \frac{d_x^3}{3}$ using Cholesky factorization [48]. Here, we assume the dimensions of $\mathbf{y}$ and $\mathbf{x}$ are d_y and d_x respectively. Note that the $\mathbf{\Phi}$ involved in matrix multiplication can be computed very fast, since every column of $\mathbf{\Phi}$ has at most only one nonzero element as mentioned in Section II and detailed in [5]. A summary of the computational costs for standard computation is listed in Scenario 1 of Table I.

Note that when the covariance matrix $\mathbf{R}$ and projection matrix $\mathbf{\Phi}$ are translation-invariant with respect to patches, $\mathbf{A}$ can be pre-computed and pre-stored. As a result, in "standard computation" section of Table I, the first row in the "Patch-dependent" part are moved to the "Patch-independent" part and the total computational cost in the "Patch-dependent" part reduces from $O(Kd_y^3/3+Kd_y^2+Kd_yd_x)$ to $O(Kd_y^2+Kd_yd_x)$, which saves a lot of computation and memory.

2) Accelerated Computation: Using the derivation in (9), we can rewrite $\tilde{\mathbf{\Sigma}}_k$ and $\tilde{\boldsymbol{\mu}}_k$ as

$$\tilde{\mathbf{\Sigma}}_k = \mathbf{E}_k \mathbf{\Lambda}_k^{\frac{1}{2}} \underbrace{[\boldsymbol{I} + \mathbf{\Lambda}_k^{\frac{1}{2}} (\mathbf{\Phi}\mathbf{E}_k)^T \boldsymbol{R}^{-1} \mathbf{\Phi}\mathbf{E}_k \mathbf{\Lambda}_k^{\frac{1}{2}}]^{-1}}_{\stackrel{\text{def}}{=} \boldsymbol{G}} \mathbf{\Lambda}_k^{\frac{1}{2}}, \quad (15)$$

$$\tilde{\boldsymbol{\mu}}_k = \mathbf{E}_k \mathbf{\Lambda}_k^{\frac{1}{2}} \underbrace{[\boldsymbol{I} + \mathbf{\Lambda}_k^{\frac{1}{2}} (\mathbf{\Phi}\mathbf{E}_k)^T \boldsymbol{R}^{-1} \mathbf{\Phi}\mathbf{E}_k \mathbf{\Lambda}_k^{\frac{1}{2}}]^{-1}}_{=\boldsymbol{G}} \mathbf{\Lambda}_k^{\frac{1}{2}} (\mathbf{\Phi}\mathbf{E}_k)^T \boldsymbol{R}^{-1} \mathbf{y}$$
$$+ \mathbf{E}_k \mathbf{\Lambda}_k^{\frac{1}{2}} \underbrace{[\boldsymbol{I} + \mathbf{\Lambda}_k^{\frac{1}{2}} (\mathbf{\Phi}\mathbf{E}_k)^T \boldsymbol{R}^{-1} \mathbf{\Phi}\mathbf{E}_k \mathbf{\Lambda}_k^{\frac{1}{2}}]^{-1}}_{=\boldsymbol{G}} \mathbf{\Lambda}_k^{-\frac{1}{2}} \mathbf{E}_k^T \boldsymbol{\mu}_k. \quad (16)$$

For convenience, denote the number of significant nonzeros eigenvalues in $\mathbf{\Lambda}_k$ by h_k ($h_k \ll d_x, d_y$).

As before, we categorize the computations into the patch-independent and patch-dependant parts. Besides the EM algorithm, the patch-independent part also includes the eigen-decomposition operation for all K Gaussian components, $\mathbf{\Sigma}_k = \mathbf{E}_k \mathbf{\Lambda}_k \mathbf{E}_k^T$, $\forall k$. In the patch-dependant part, though computations in (6), (15) and (16) involve matrix inversion $\boldsymbol{G}$, the associated computational cost is significantly reduced as h_k (the number of significant eigenvalues in $\mathbf{\Lambda}_k$) is much smaller than d_x and d_y. A summary of the computational costs for accelerated computation is listed in Scenario 2 of Table I. Similar to standard computation, when $\mathbf{R}$ and $\mathbf{\Phi}$ are translation-invariant with respect to patches, $\boldsymbol{G}$ can be pre-computed and pre-stored. Consequently, the total computational cost in the "Patch-dependent" part in Scenario 2 reduces from $O(Kh_k^3/3+Kh_kd_x+Kh_kd_y)$ to $O(Kh_kd_x+Kh_kd_y)$.

IV. ADAPTIVE INTEGRATION WINDOW

With compressive video cameras of the type in [3, 4, 5], it is relatively straightforward to adjust the integration window Δ_t. For a desired reconstruction fidelity, the maximum allowable Δ_t is related to the complexity ($e.g.$, velocity) in the associated motion within the scene. For periods of time with minimal temporal variation, Δ_t may be made large, as there will not be substantial blur manifested in summing many frames over the window Δ_t. However, when the motion within the scene is substantial, it is desirable to make Δ_t smaller. The value of Δ_t defines the degree of compression, and therefore it is desirable to adapt Δ_t based upon the complexity of the scene under test ($i.e.$, to make Δ_t as large as possible, within a desired reconstruction-quality constraint).

Recall that T frames are recovered within time window Δ_t. We assume sampling rate δ_t between consecutive frames, and therefore $\Delta_t = T\delta_t$. Within the CACTI system, δ_t is only employed within the software-based inversion for the underlying video, and therefore variable δ_t may be examined within the context of inversion, given compressive video measurements. The mask in CACTI moves continuously, and therefore δ_t is set with the goal of making the mathematics of the inversion accurate (the mask is approximately fixed in space over time δ_t); by contrast, in [3, 4] δ_t is set by the hardware. We here assume δ_t is fixed, and then by varying the integration window Δ_t we also change the number of recovered frames, $T = \Delta_t/\delta_t$, where T is assumed to be an integer.

Using the video CS camera of interest, here CACTI, we may constitute *simulated* compressive measurements, for variable Δ_t. Since the true underlying video is known, we may compute the quality of the reconstruction ($e.g.$, peak signal to noise ratio (PSNR)) as a function of Δ_t. With a specified desired reconstruction quality, for each training video example we may specify the maximum associated Δ_t. Simulated data and associated reconstructions may be considered for many types of scenes, and for each a range of Δ_t are considered; for each case the maximum Δ_t is selected within the fidelity criterion.

After this process, we have access to a set of video clips, of variable length Δ_t, and compressive measurements over that window yielded the specified reconstruction fidelity. We would like to use this training data to design a means by which given compressive video, we may assign the appropriate compressive

integration window Δ_t. Toward that end, we extract features from these video clips, and learn a mapping between the features and the associated appropriate compressive measurement window Δ_t. Once such a function is so learned, one may compute feature vectors "on the fly" based upon recent recovered video (after CS inversion), and map this feature vector to an associated updated integration window Δ_t.

We estimate the degree of motion within each video clip using the block-matching method, which has been employed in a variety of video codes ranging from MPEG1/H.261 to MPEG4/H.263 [49, 50]; the corresponding hardware has been well developed, and hence it may be done quickly in practice. Within each video clip we compute the maximum velocity within the scene. We then employ a standard Support Vector Machine (SVM) [51, 52] to map this feature (velocity) to Δ_t. To simplify the analysis, we consider a library of Q possible values for Δ_t, and the SVM does a multi-class mapping from the maximum velocity to the associated Δ_t within the Q-dimensional library. The use of a finite library of values of Δ_t is expected to simplify practical implementation, and it also simplifies inversion, as GMM signal models are only required for this finite set of Q integration windows Δ_t.

Within the experiments, the maximum velocity is computed based on the previous time window of length Δ_t, and this is then used to define the Δ_t in the next time window of compressive measurements. Of course, in practice, the rate at which Δ_t is adjusted is defined by hardware considerations, and the speed with which a method like the SVM can define an update for Δ_t.

In the above procedure, we define Δ_t in terms of the maximum velocity *anywhere* in the video clip. However, in many situations, the moving objects within the scene are localized. This suggest that the value of Δ_t may vary as a function of spatial subregion within the scene. This can be done in principle, but it complicates camera implementation, and therefore it is not considered here. Another way is to directly estimate the maximum of velocity from the measurements instead of from the reconstructed video clips [53]. Though motion information may not be well maintained in the measurements, the online adaptivity of Δ_t becomes feasible.

V. Experimental Results

We demonstrate the proposed reconstruction method on a variety of scenes and present results in comparison with current leading reconstruction methods. All associated videos in the experiments can be accessed at http://people.duke.edu/~jy118/CACTI_GMM.htm.

A. Experimental Settings

For the proposed GMM-based inversion method, each coded measurement of size $N_x \times N_y$ is partitioned into a collection of overlapping patches of size $P \times P$, by horizontally and vertically sliding a $P \times P$ block for v pixels, $v \in [1, 2 \cdots, P]$. Accordingly, the number of patches ranges from $(N_x - P + 1) \times (N_y - P + 1)$ to $(N_x/P) \times (N_y/P)$ as v changes from 1 to P. To balance simplicity and accuracy, we set $v = P/2$ and $P = 8$ throughout all experiments.

Regarding the measurement model, covariance matrix $\boldsymbol{R}$ for Gaussian distribution $p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \boldsymbol{R})$ is assumed to be a scaled identity matrix $\boldsymbol{R} = \sigma^2 \boldsymbol{I}$ with $\sigma = 10^{-3}$. In the GMM, the number of components K is simply fixed to 20 (although in practice fewer than K components could be used). To avoid ~~the~~ singularity of the covariance matrices, we add ~~a~~ ε to the diagonal of every covariance matrix to ensure that the covariance matrices are positive definite. Following the strategy presented in [54] (Chapter 9), we initialize the GMM in the EM algorithm as follows: we first adopt the k-means algorithm to cluster the training data, and then use the means of the clusters to initialize $\{\boldsymbol{\mu}_k\}$, the sample covariances of the clusters to initialize $\{\boldsymbol{\Sigma}_k\}$ and the fractions of the training patches assigned to the respective clusters to initialize $\{\boldsymbol{\pi}_k\}$. To balance efficiency and effectiveness of the EM algorithm, the number of training patches is ~~set in the range of 20,000 to 100,000~~ depending on the dimension of signals. In the online-updated GMM algorithm, the decay-rate parameter κ is set to 2.

For synthetic data, we use PSNR and SSIM [55] to evaluate the quality of every reconstructed frame, and finally the average values over all video frames is taken as the evaluation metric for video reconstruction. Mean square error (MSE) has also been used in evaluation, and it reflects nearly same patterns as PSNR in our experiments. Hence, the results on MSE are not reported. For real data, where ground truth is not available, the (subjective) visual quality is used to evaluate reconstruction.

B. Comparison with Other Methods

The proposed GMM-based inversion methods are compared with the following leading reconstruction methods:

• Generalized Alternating Projection (GAP) [56][1]. GAP is a generalized alternating projection method that solves the minimization problem:

$$\min_{\boldsymbol{f}} \|\boldsymbol{f}\|_{\ell_{2,1}^{\mathcal{G}\beta}}, \quad \text{s.t. } \mathbf{H}\boldsymbol{W}\boldsymbol{f} = \text{vec}(\mathbf{Y}), \tag{17}$$

where $\|\boldsymbol{f}\|_{\ell_{2,1}^{\mathcal{G}\beta}} = \sum_{k=1}^{s} \beta_k \sqrt{\sum_{i \in \mathcal{G}_k} f_i^2}$, $\text{vec}(\cdot)$ vectorizes a matrix inside $()$, $\mathbf{H}$ is CACTI-specified projection matrix expressed as $\mathbf{H} \overset{\text{def}}{=} [\mathbf{H}_1 \ \mathbf{H}_2 \cdots \ \mathbf{H}_T]$ with $\mathbf{H}_t \overset{\text{def}}{=} \text{diag}[\mathcal{A}_{1,1,t} \ \mathcal{A}_{2,1,t} \ \cdots \mathcal{A}_{N_x,N_y,t}]$, $t = 1, \ldots, T$, and $\boldsymbol{W}$ is the domain transform matrix that is imposed on the video signal $\mathbf{X}$ to achieve the sparse representation by $\boldsymbol{f}$, *i.e.*, $\mathbf{X} = \boldsymbol{W}\boldsymbol{f}$. In (17), $\boldsymbol{f}$ is partitioned into s non-overlapping groups $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_s\}$, and each group $\mathcal{G}_k$ is associated with a constant weight β_k, $k = 1, \ldots, s$. In the experiments, DCT and wavelet transform are used to yield sparse representation for $\mathbf{X}$, and the default settings on $\beta_k, \mathcal{G}_k, \ \forall k$ are employed.

• Two-step iterative shrinkage/thresholding (TwIST) [57][2]. TwIST solves the unconstrained optimization problem:

$$\min_{\mathbf{X}} \|\text{vec}(\mathbf{Y}) - \mathbf{H}\text{vec}(\mathbf{X})\|^2 + \tau\boldsymbol{\Omega}(\mathbf{X}) \tag{18}$$

[1]The source code of GAP is available upon request.

[2]The source code of TwIST can be downloaded from http://www.lx.it.pt/~bioucas/code.htm .

Fig. 2. Example frames of the unrelated offline training videos used in the proposed GMM-based inversion method and KSVD-OMP method.

where $\mathbf{\Omega}(\mathbf{X})$ is the regularizer and τ is the parameter to balance the ℓ_2-loss term and the regularizer term. Following previous CS inversion applied to still images and hyperspectral images [58], the generalized 2D total variation (TV) is used as the regularizer:

$$\mathbf{\Omega}(\mathbf{X}) = \sum_{n} \sum_{i,j} \left\{ (\mathbf{X}_{i+1,j,n} - \mathbf{X}_{i,j,n})^2 + (\mathbf{X}_{i,j+1,n} - \mathbf{X}_{i,j,n})^2 \right\}^{\frac{1}{2}}.$$

The results of TwIST reported here represent the best among the different settings of τ in the range of $[10^{-4}, 1]$.

In addition, we have also consider the 3D TV,

$$\mathbf{\Omega}(\mathbf{X}) = \sum_{i,j,n} \left\{ (\mathbf{X}_{i+1,j,n} - \mathbf{X}_{i,j,n})^2 + (\mathbf{X}_{i,j+1,n} - \mathbf{X}_{i,j,n})^2 + (\mathbf{X}_{i,j,n+1} - \mathbf{X}_{i,j,n})^2 \right\}^{\frac{1}{2}},$$

as a regularizer for TwIST, and included it in the experiments. However, TwIST with the 3D TV performs slightly worse than its 2D counterpart, and therefore are not reported here.

• KSVD and orthogonal matching pursuit (KSVD-OMP) [40, 59][3]. The state-of-the-art over-complete dictionary learning method KSVD [40] with the sparse signal estimation method OMP [59] is used as the third baseline. This method has been widely used various CS applications, like still image CS [6], hyperspectral image CS [58][4], and video CS [4]. As mentioned in Section III-C, KSVD learns the over-complete dictionary $\boldsymbol{W}$ from the video patches obtained from the training videos, and OMP obtains the sparse representation $\boldsymbol{f}$ for signal $\mathbf{x}$. The size of a video patch is $8 \times 8 \times T$. For the sake of fair comparison, KSVD uses the same training videos as the proposed GMM method. The number of dictionary elements is tuned among the six possible values within $\{128, 256, 512, 1000, 3000, 5000\}$, and the best results are shown in the experiments.

C. Results on Synthetic Data: Unrelated Training Data

We first present results on synthetic data. Following the coding mechanisms in CACTI, each frame of video to be reconstructed is encoded with a shifted binary mask, that is simulated by a random binary matrix with elements drawn from a Bernoulli distribution with parameter 0.5. The measurement is constituted by summing T such coded frames. Note that in these simulations the mask is exactly fixed for each of these T time points, while in the real camera this is an approximation. A zero-mean white noise with standard devaluation $\sigma = 10^{-3}$ is added to the simulated measurement, to represent model mismatch in the actual (real) data.

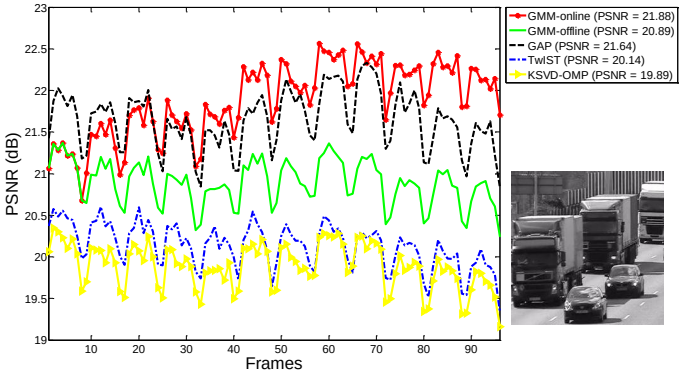


Fig. 3. PSNR comparison of the offline and online GMM, TwIST [57], GAP [56] and KSVD-OMP [40, 59] for the `traffic` dataset ($T = 8$). The PSNR values in the bracket in the upper-right corner are the average PSNR over all 96 frames. The corresponding reconstructed frames are displayed in Fig. 5.

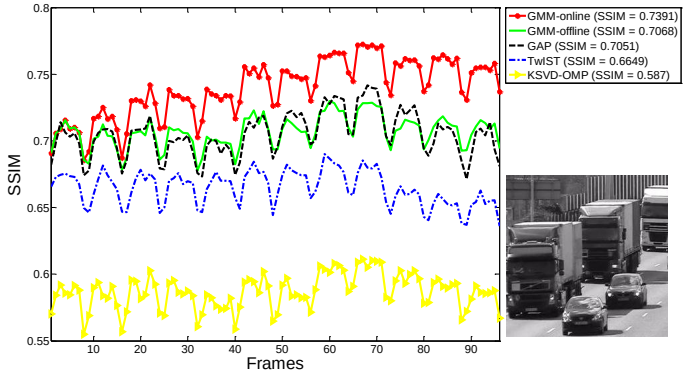


Fig. 4. SSIM comparison of the offline and online GMM, TwIST, GAP and KSVD-OMP for the `traffic` dataset ($T = 8$). The SSIM values in the bracket in the upper-right corner are the average SSIM over all 96 frames.

We synthesize three videos for reconstruction. They contain scenes denoted `traffic`, `couple`, and `train`. These three synthetic videos have different characteristics. The `traffic` and `train` are dominated by a moving foreground, while the `couple` video has more of a fixed background. The `traffic` video contains multiple moving objects, while the `train` video contains one moving object. All synthetic videos are of size $256 \times 256 \times 96$, and have temporal compression ratio $T = 8$.

For the proposed GMM-based inversion method, three-dimensional video patches are of size $8 \times 8 \times T$, *i.e.*, $T_0 = T$, and offline-training data are twenty arbitrary (distinct) videos, which contain the various scenes and object motion (see

[3]The source code of KSVD-OMP can be downloaded from http://www.cs.technion.ac.il/~ronrubin/software.html

[4]In [6, 58], KSVD-OMP serves as the baseline for their proposed CS inversion algorithm.

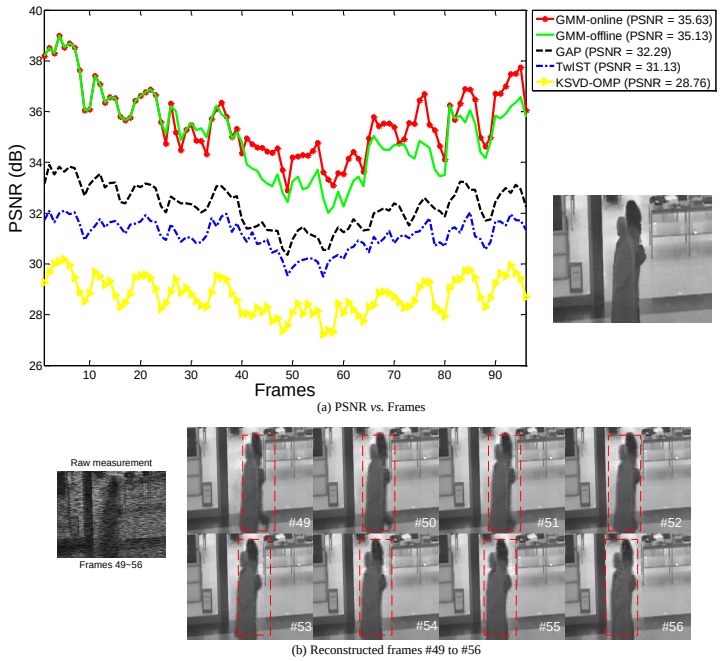
Fig. 5. Reconstructed frames 89 to 96 for the `traffic` dataset ($T = 8$) by the offline and online-updated GMM, TwIST, GAP and KSVD-OMP methods.

example frames in Fig. 2)[5], respectively. Note these training videos are all unrelated to the videos to be reconstructed.

1) Reconstruction Results: We plot PSNR curves to compare the reconstruction performance quantitatively, and also display the reconstructed video of selected video frames for psychovisual comparison. These results are shown in Figs. 3, 6 and 7 for the data `traffic`, `couple` and `train`, respectively. Due to the space limitations, reconstructed frames by GAP, TwIST and KSVD-OMP for the `couple` and reconstructed frames by all methods for `train` are not shown here. Interested readers may refer to our aforementioned website for all the videos and frames. From the PSNR results shown in these figures, the online-updated GMM consistently performs best among all methods for these three synthetic datasets. Importantly, its PSNR is generally increasing as more frames are reconstructed. This vividly illustrates the merit of online learning. Fig. 4 plots the SSIM curves for the `traffic` dataset. The results in this figure show similar pattern to the PSNR results in Fig. 3, and further verify the performance advantage of the proposed method over the other reconstruction methods. This kind of observations are also observed in the `couple` and `train` datasets, and SSIM curves are hence not shown for these datasets.

The GAP algorithm generally performs best among all baselines. KSVD-OMP generally performs worse among all methods. Visual quality evaluations on the reconstructed frames are consistent with the PSNR and SSIM results. Specifically, from Fig. 5, the reconstructed frames by TwIST and KSVD-OMP have the bold artifacts among both foreground and background, while reconstructed frames by GAP look better but with more blurring. The reconstructed frames by the offline GMM method looks well on the background but some detailed information on the foreground are not well maintained. The online-updated GMM can generally reconstruct the frames with sharper scenes than other methods (notice the small moving vehicles).

[5]The training videos can be downloaded from http://projects.cwi.nl/dyntex/database_pro.html, http://www.di.ens.fr/ laptev/actions/, http://www.brl.ntt.co.jp/people/akisato/saliency3.html



(a) PSNR *vs.* Frames

(b) Reconstructed frames #49 to #56

Fig. 6. (a). PSNR comparison of the offline and online GMM, TwIST, GAP and KSVD-OMP for the `couple` dataset ($T = 8$). (b). Reconstructed frames 49 to 56 by the online-updated GMM method.

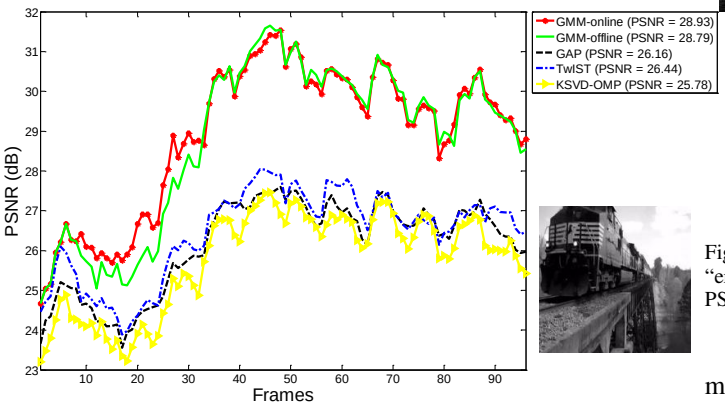


Fig. 7. PSNR comparison of the offline and online GMM, TwIST, GAP and KSVD-OMP for the `train` dataset ($T = 8$).

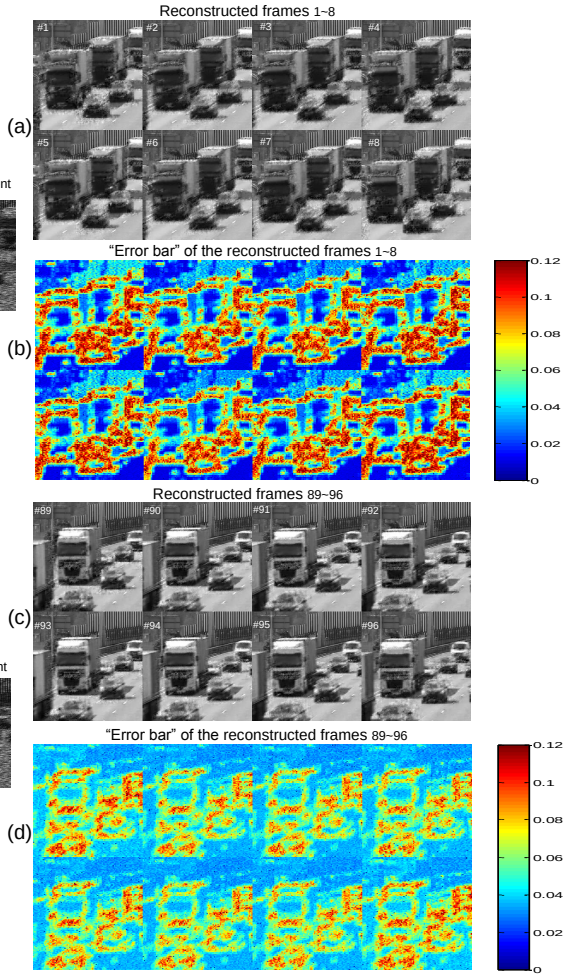


Fig. 8. Reconstruction frames 1-8 (a), 89-96 (c) and their corresponding "error bars" (b,d) for the `traffic` dataset. Fig. 3 shows the corresponding PSNR values frame by frame.

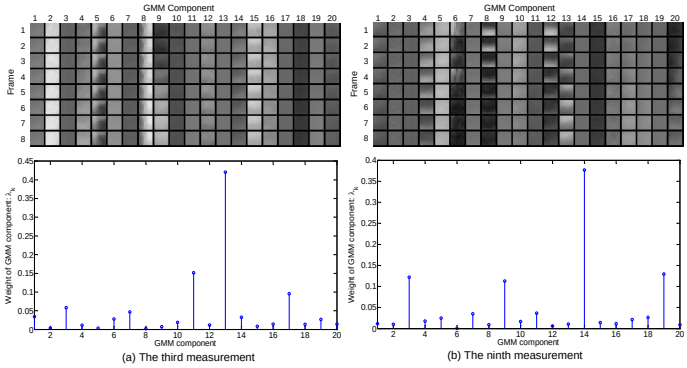
In addition, compared with GAP, TwIST and KSVD-OMP, the proposed GMM method can provide richer probabilistic information, such as confidence of reconstruction, or "error bars". Specifically, the "error bar" is defined as the square root of variance of the estimator. In the proposed method, covariance of the estimator, denoted by $\mathrm{Cov}(\mathbf{x}_i|\mathbf{y}_i)$, is expressed below:

$$\begin{aligned}\mathrm{Cov}(\mathbf{x}_i|\mathbf{y}_i) &= \int (\mathbf{x}_i - \mathbb{E}(\mathbf{x}_i|\mathbf{y}_i))(\mathbf{x}_i - \mathbb{E}(\mathbf{x}_i|\mathbf{y}_i))' p(\mathbf{x}_i|\mathbf{y}_i)\mathrm{d}\mathbf{x}_i \\ &= \sum_{k=1}^{K} \tilde{\lambda}_k \left\{ \tilde{\mathbf{\Sigma}}_k + (\tilde{\mathbf{x}}_i - \tilde{\boldsymbol{\mu}}_k)(\tilde{\mathbf{x}}_i - \tilde{\boldsymbol{\mu}}_k)' \right\},\end{aligned}$$

where $p(\mathbf{x}_i|\mathbf{y}_i)$ and $\mathbb{E}(\mathbf{x}_i|\mathbf{y}_i)$ are given by (5) and (8), respectively. The diagonal elements of $\mathrm{Cov}(\mathbf{x}_i|\mathbf{y}_i)$ are taken as the variance of the estimator. Fig. 8 shows the error bars of reconstruction provided by the online-updated GMM method for the `traffic` dataset, at frames 1-8 and 89-96. From the figure, frames 89-96 not only have the better visual quality than frames 1-8, but also have lower error bars. This implies that as the inversion algorithm proceeds, the confidence of reconstruction increases. This again validates the efficacy of the online-updated GMM method. In addition, at each reconstructed frame, the foreground often corresponds to high "error bar" relative to the background. This is reasonable, as the foreground often has lower temporal redundancy.

2) Investigation on updated prior, soft-assignment property and patch size: In the online-updated GMM, the parameters of the GMM are updated once within every time integration window $(l-1)T_0 < t \leq lT_0$ $(\forall l = 1, 2, \dots)$. Fig. 9 demonstrates the updated priors at two integration windows. From the figure, it is clear that the patterns of the updated GMM components and the component weights are significantly different after updating.

Fig. 10 illustrate the soft-assignment property (as mentioned in Section III-C2) of the proposed GMM-based inversion algorithm. Recall that the posterior distribution is also a mixture of Gaussian distribution as derived in (5). We perform the offline GMM-based inversion algorithm on the `traffic` dataset



(a) The third measurement (b) The ninth measurement

Fig. 9. Illustration of the online-updated GMM priors for the first measurement ($0 < t \leq T_0$)and the ninth measurement ($8T_0 < t \leq 9T_0$), respectively. In each subfigure, the upper part shows all Gaussian components in 20 columns and each Gaussian component is shown in 8 (T_0) image patches size of 8×8 ($P \times P$), and the lower part shows the weights of all Gaussian components.

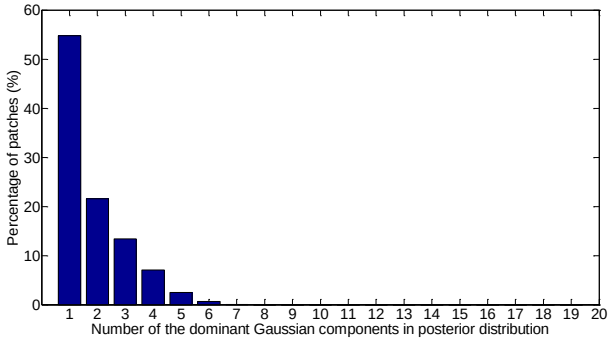


Fig. 10. Illustration of the soft-assignment property of the proposed method. The `traffic` dataset for the first measurement is used with $T = 8$. The size of a video patch is $8 \times 8 \times T$. The Gaussian component with the weight (defined in (6)) greater than 10^{-3} are considered as a dominated components. The percentage of patches is zero when the number of the dominant Gaussian components in posterior distribution is greater than 7.

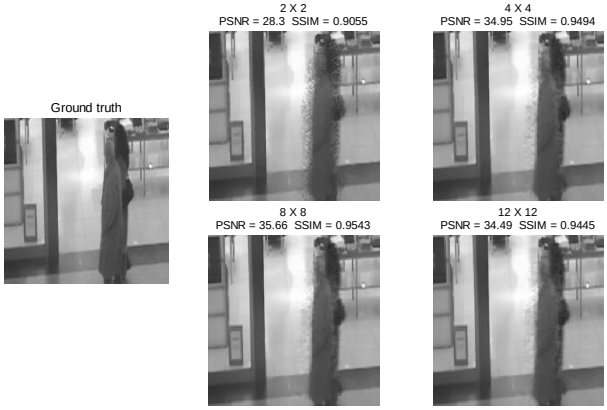
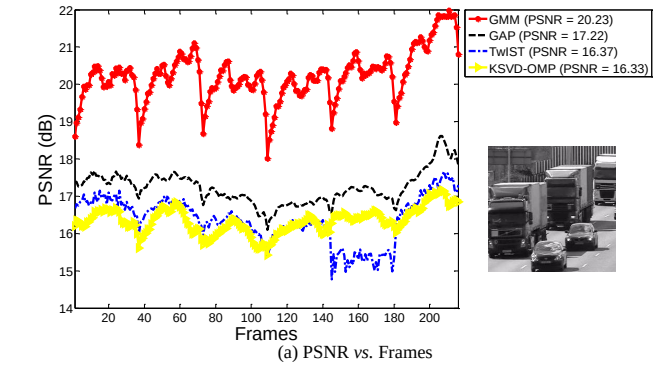


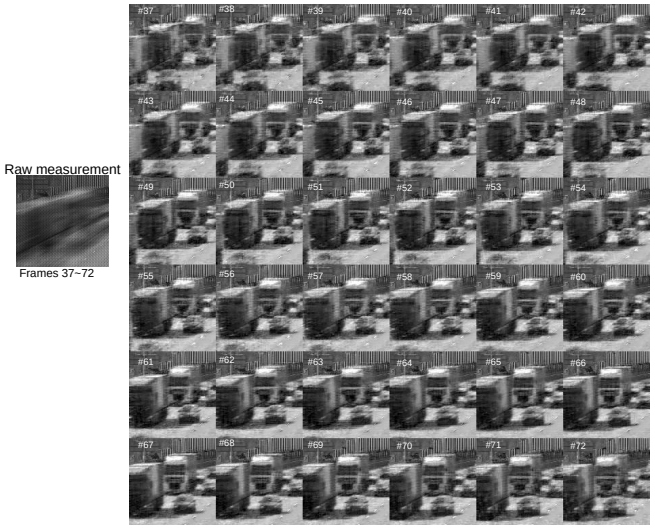
Fig. 11. Reconstruction results of `couple` dataset with $T = 8$ based on four different patch size. The second frame out of eight video frames are shown.

within one integration window. From this figure, we found that more than 40% of patches have a posterior distribution with more than one dominated Gaussian components. This justifies the soft assignment implied in the conditional expectation (8).

The effect of different choices of patch size in the GMM-based algorithm to the video reconstruction fidelity is investigated on the `couple` dataset with $T = 8$. The four different settings of patch size are compared in the offline GMM



(a) PSNR *vs.* Frames



(b) Reconstructed frames #37 to #72

Fig. 12. (a). PSNR comparison of the offline-GMM, TwIST, GAP and KSVD-OMP for the `train` dataset ($T = 36$). (b). The reconstructed frames 37-72 by the online-updated GMM method.

method. We can see that although the inversion algorithm with all these four settings of patch size can yield the reasonable reconstruction performance, the patch size 8×8 generally leads to the sharper reconstructed frames and less artifacts on the moving objects. This is confirmed by the quantitative values of the PSNR and SSIM metrics. In terms of computational cost, since the dimension of signals d_x quadratically increases with respect to the patch size, larger d_x can significantly increase the costs of EM and inversion procedures as analyzed in Section III-D. To balance the performance and computational cost, we choose the patch size 8×8 for the proposed method.

D. Results on Synthetic Data: Related Training Data

We now evaluate the three reconstruction methods on two challenging tasks: high compression ratio and complicated motion. The first video is the same `traffic` dataset used above, but with higher compression ratio $T = 36$. The second video contains the `NBA game` scene, with $T = 8$, where a player has complicated movement during his dunk. These two videos are of size $256 \times 256 \times 216$ and $256 \times 256 \times 96$, respectively. For the proposed GMM method, we employ related data to train the GMM. Specifically, for the `traffic` video, the related training data are traffic videos having the

same type of background but different foreground (moving vehicle). For the NBA game video, the related training data are videos containing similar NBA games (they generally do not contain the same background). It is worth noting that utilizing related training data is often reasonable, since in some cases, like surveillance in the highway, we can first record some uncompressed videos and then use these videos as the training data to reconstruct the upcoming compressed videos. As the related training data are with high visual quality and strong relevance, the trained GMM is deemed to be good enough such that the online-updated GMM is not necessary in general. This has been verified in our extensive experiments. Thus, we adopt the offline-trained GMM to directly reconstruct the above two challenging videos; three-dimensional video patches of size $8 \times 8 \times T$ are used.

Fig. 12 plots the PSNR curves against frame for the GMM, GAP, TwIST and KSVD-OMP methods, for the highly-compressed traffic data, and displays the reconstructed frames 37 to 72 (corresponding to the second compressive measurement) by the proposed method. The reconstructed frames by GAP, TwIST and KSVD-OMP are not shown here due to space limitations. From the figure, the proposed GMM method performs significantly better than GAP, TwIST and KSVD-OMP in terms of PSNR and visual quality (please refer to our website). As far as the proposed method is concerned, its average PSNR value decreases 1.6dB compared with the results in the case of $T = 8$ (shown in Fig. 3), while the reconstructed frames for $T = 36$ still maintain nearly comparable visual quality as those with $T = 8$. Note that GAP and TwIST do not utilize training data at all, and their generality is therefore undermined in this specific video challenge. On the other hand, the GMM results could be improved even further via online adaptivity to the video under test.

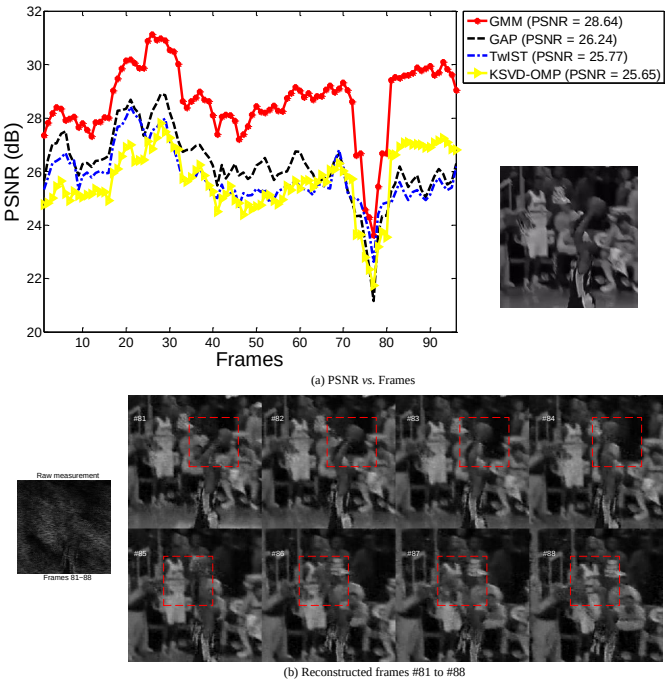
Fig. 13 plots PSNR for the GMM, GAP, TwIST and KSVD-OMP methods, for the NBA game dataset, and also display the reconstructed frames 81 to 88 (corresponding to the eleventh measurement) by the proposed method. From the figure, the proposed GMM method improves upon GAP, TwIST and KSVD-OMP by 2.40 dB, 2.87 dB and 2.99 dB on average, respectively. Importantly, all methods have PSNRs that decrease at frames 70 to 80, which correspond to the moment of the dunk. However, the proposed method still performs better than GAP, TwIST and KSVD-OMP in this period. From the reconstructed frames, the proposed method can provide high visual quality (of course, this is subjective). Compared with the reconstructed frames by GAP, TwIST and KSVD-OMP (please refer to our website), the proposed method maintains the highest fidelity in the foreground (the regions around the ball and the player). Note that defocusing exists in the ground truth (notice the audience in the video) and offline training data. As a consequence, the blur among audiences exists in the reconstructed frames for all reconstruction methods.

E. Results on Real CACTI Data

We now present results to demonstrate the efficacy of the proposed method on *real* data acquired by the CACTI camera we have built. This is a significant contribution, as the GMM-based inversion method has not been applied to reconstruct the videos for real data so far. As an effective reconstruction approach, in the following, we show the proposed method outperforms the GAP, TwIST and KSVD-OMP methods for three real datasets: chopper wheel, eye blink and hand lens. The chopper wheel dataset contains the scene that a letter "D" is placed at the edge of a chopper wheel. The wheel is rotating in front of the camera at an angular velocity of 15 blades per second. The eye blink dataset contains the scene of an eye blink from closed to open. The hand lens dataset contains the scene of an optical lens falling in front of a hand. These three videos are composed of 6, 8 and 5 compressed measurements, respectively. The CACTI camera captures data at 30 frames per second (fps). The integration window Δ_t for each measurement is 33ms. The number of reconstructed frames from a single compressive measurement is $T = 14$, so each frame corresponds to $\delta_t = 2.36$ ms (420 fps). The measurement matrix $\mathbf{\Phi}$ is provided by the details of the CACTI hardware.

The reconstruction results of these three datasets using the proposed GMM method are displayed in Fig. 14, along with the TwIST, GAP and KSVD-OMP results[6]. The reconstructed frames by the three methods correspond to the sixth, fifth and third measurements for the chopper wheel, hand lens and eye blink datasets, respectively. Due to the difficulty of regenerating the temporal motion (the motion only lasts around 0.15s in real life), the reconstructions on these three datasets are evaluated solely by visual quality. Specifically, for chopper wheel, the online-updated GMM provides the clearest character "D" among the four methods. The character "D" exhibits ghosting due to ambiguities within the reconstructed solutions. While this ghosting effect exists for all three methods, the proposed method exhibits little ghosting and better preserves the video clarity. For eye blink, all results can successfully show the process of the eye blink, from closed to open, while the proposed method appears to provide more complete process of the eye blink. For hand lens, all methods can recover the falling process of the lens. TwIST does have clearer background than the other results. However, the artifact in the simulation results now can also be observed from the red-box marked in frame 1. Only the GMM method can provide the unbroken edge (the small hole at right-bottom inside the red-box) of the lens. GAP still has the obvious blurring effect, which can be noticed from the red-box marked in frame 14.

[6]The related videos are adopted to train the GMM prior and the dictionary for the proposed method and KSVD-OMP method. For the chopper wheel and hand lens datasets, the training videos were captured with a camcorder prior to being adjusted to the desired framerate of the decompressed videos. For the chopper wheel dataset, the training videos include a chopper wheel rotating at several orientations, positions, and velocities. For the hand lens dataset, the training videos include a lens falling in front of a hand at different velocities. For the eye blink data, the training videos include related eye blink videos that are downloaded from website http://www.fotosearch.com/photos-images/eye-video.html. The three-dimensional video patch used in GMM is of size $8 \times 8 \times 14$ for the chopper wheel and the hand lens data, and $8 \times 8 \times 28$ for the eye blink dataset.



(a) PSNR *vs.* Frames

(b) Reconstructed frames #81 to #88

Fig. 13. (a) PSNR comparison of the offline GMM, TwIST, GAP and KSVD-OMP for the `NBA game` dataset ($T = 8$). (b) The reconstructed frames 81-88 by the online-updated GMM method.

F. Performance Comparison under Other Coding Strategies

The proposed GMM algorithm can be used for a video CS camera using any local coding strategy [7]. In this section, we consider the coding strategies in [3, 4] as well as a more general strategy, and compare the proposed algorithm to other video inversion algorithms in these cases. The DMD/LCoS modulator in [3] is simulated as random binary matrices with entries i.i.d. from a Bernoulli distribution with parameter 0.5, each matrix applied to a distinct frame. The bump modulator in [4] is simulated as a sequence of dependent binary random matrices, each for a distinct frame, with the dependency being such that each pixel is turned on for b consecutive time steps, starting from a random time drawn from $[1, 2, \ldots, T-b+1]$, where T equals the total number of steps during the temporal integration. In addition, we also consider a more general coding strategy in CACTI, in which the mask is simulated by a random matrix with each element independently drawn from a uniform distribution in $[0, 1]$. Lastly, we consider the case in which the elements of the mask are drawn from a normal distribution with mean as 0.5 and standard deviation as 1. Considering the hardware constraints, we truncate the elements of mask within $[0, 1]$. Fig. 15 shows the reconstruction results on the `Traffic` dataset ($T = 8$) by various methods, under the above four coding strategies. It is seen that all methods yield similar performances as those in Fig. 3, and the proposed online adapted GMM method consistently performs the best under all coding strategies. These results demonstrate the efficacy of the proposed method under general coding strategies.

[7]The proposed method is a patch-based model and cannot work with global measurement matrices as those used on a single-pixel camera [13, 11].

G. Adaptive Integrative Window

We evaluate the proposed adaptive integration window Δ_t on the simulated `traffic` data. Assuming δ_t is fixed, the adaptive integration window Δ_t is equivalent to adapting the number of encoded frames T in a measurement. The offline-trained GMM is used and the video patch is of size $8 \times 8 \times 12$. We begin the experiment with $T = 6$. As time progresses, the proposed method varies T according to the estimated motion within each integration window. We artificially change the speed of the foreground during the 168 frames comprising this video to evaluate the fidelity of the proposed method. Frames 1-60 (Fig. 16(b)) and 121-168 (Fig. 16(d)) play at the originally-captured frame rate, while the scene is frozen between frames 61-120 (Fig. 16(c)).

The proposed GMM updates T every 12 reconstructed frames (Fig. 16), as the same size of video patch is constantly used. From Fig. 16, we observe that the proposed method can effectively adapt T based on the estimated motion velocity in the reconstructed video. Specifically, the algorithm holds T at the initial value 6 for the first 72 frames, before increasing T to 12 when the scene motion stops. After the scene recovers to play at the originally-captured frame rate, T drops from 12 to 6. There are 12-frame delay of adapting T when the velocity is changed. This is reasonable, as the proposed method uses the reconstructed frames in the last T_0 frames to predict T for the current T_0 frames as mentioned in Section IV.

The average T over all frames in the proposed method is 8.14. We also evaluate the performance of the reconstruction without adaptive T, *i.e.*, $T = 8$. The average PSNR values for the adaptive T and the fixed T are 28.73dB and 26.31dB, respectively.

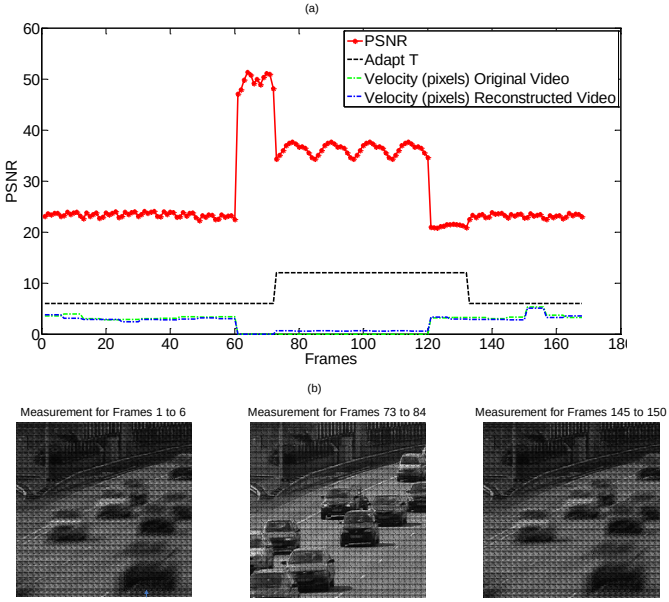


Fig. 16. Demonstration of adaptive temporal compression ratio on synthetic traffic data: (a) the plots of PSNR, adaptive CS ratio T, estimated velocity from original videos and reconstructed videos against frame; (b) three typical measurements corresponding to different vehicle speeds.

H. Computation Time

The proposed method and alternative methods are implemented in Matlab 2012(b), and the experiments are conducted

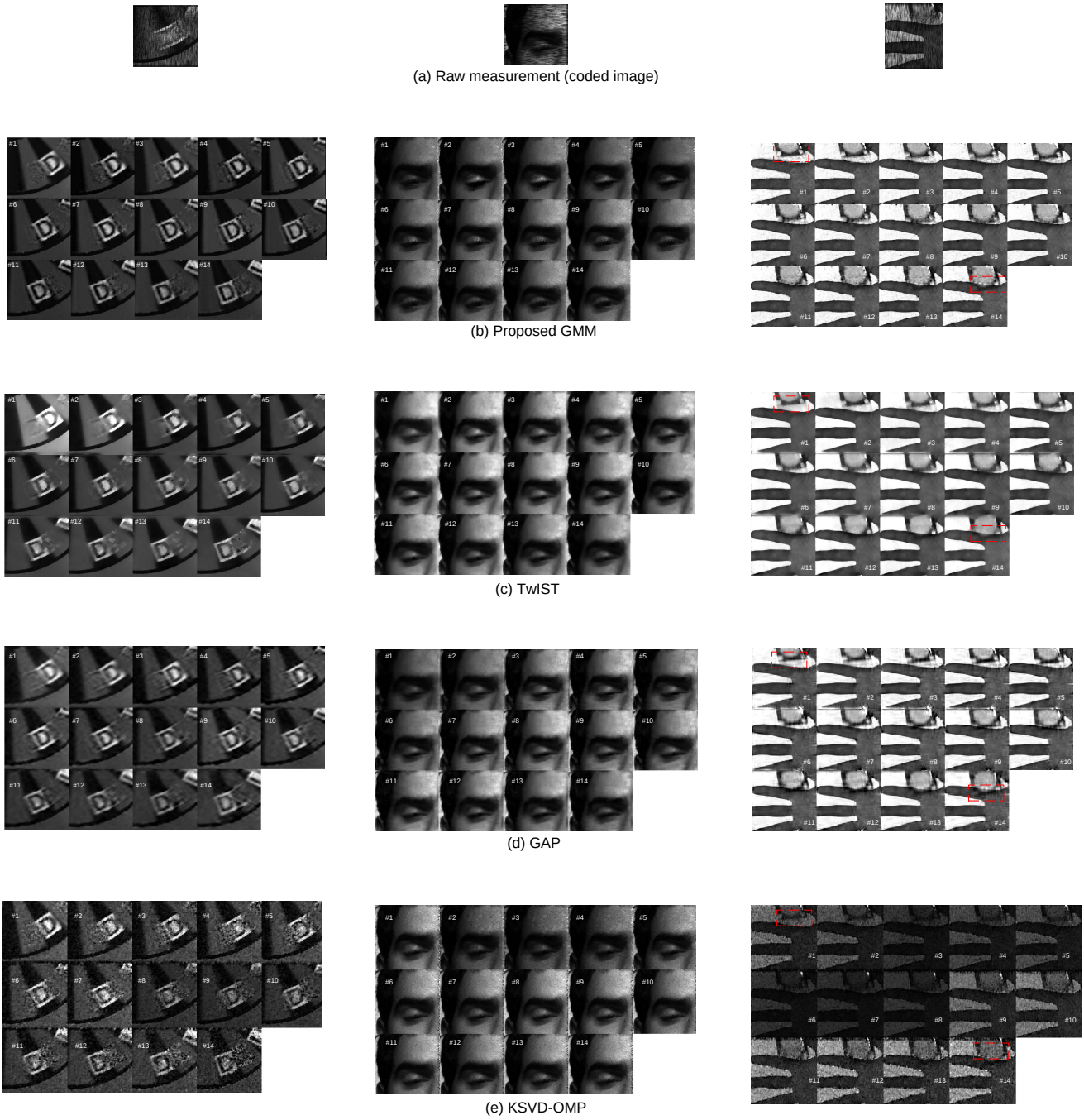


Fig. 14. (a) the raw measurements of `chopper wheel`, `eye blink` and `hand lens` dataset ($T = 14$) acquired from CACTI. The reconstructed frames from the raw measurement (a) by the GMM (b), TwIST (c), GAP (d) and KSVD-OMP (e).

on a PC with an Intel i5-2500 CPU running at 3.30 GHz and with 16GB RAM. The `chopper wheel` dataset is used to compare the running times of all methods. Specifically, we record the running time of reconstruction from one measurement by each method. Decomposing the spatial domain into 3936 overlapping patches, the proposed GMM-based method reconstructs the patches independently, each using around 0.08 seconds. TwIST and GAP spend 42 and 2 seconds, respectively, for reconstruction from one measurement. A comparison shows that the proposed method is computationally not as efficient as TwIST or GAP.

However, a prominent advantage of our method is that it reconstructs the patches independently. Therefore the computation can be achieved in parallel with respect to the batches, which provides a potential p-times speedup where p is equal to the number of patches (3936 in our case here). TwIST and GAP may also accelerate their computation by exploiting parallel computation, though the parallelization is not necessarily with respect to patches. The examination of patch-based parallel computation (as used by the proposed method) versus other parallel computation methods (as may be used by TwIST or GAP) is left to our future work. As detailed in Section III-D,

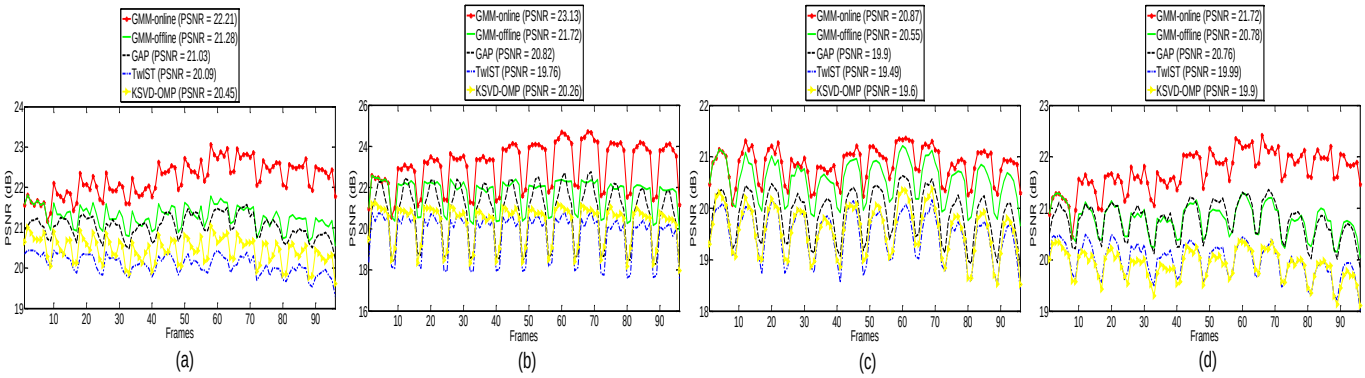


Fig. 15. Demonstrations of various methods on four different coding strategies: (a) random binary code [3]; (b) random binary code with the single bump constraint [4] (the bump length is fixed to 2 for all pixels); (c) random gray-scale code; (d) random code with elements drawn from a standard normal distribution. The `traffic` data is used with $T = 8$. The size of a video patch in GMM and KSVD-OMP is $8 \times 8 \times T$.

the computational cost of the proposed method can be further reduced if the projection matrix $\boldsymbol{\Phi}$ and covariance matrix $\boldsymbol{R}$ are translation-invariant with respect to the patches.

VI. Conclusions

A GMM-based inversion algorithm is proposed to reconstruct high-speed videos from temporally compressed measurements. Extensive experimental results show that the proposed GMM-based inversion method can reconstruct video frames with greater fidelity than other methods, on both synthetic and real data. The proposed inversion algorithm enjoys the following advantages: i) *Online*-updated GMM improves the reconstructed video quality. Following this, the reconstruction becomes less-dependent on the offline training videos and more robust to different scenes and motions. Reconstructions of both real and synthetic coded measurements show considerable improvement for the online-updated GMM method. We found that for simple motions, the online-updated GMM method with even unrelated training videos can yield good reconstruction results. On the other hand, for fast and complicated motions, the offline-learned GMM with related training videos can yield promising results. In this case, the online-updated GMM is often unnecessary. ii) *Parallel reconstruction* using a GPU is applicable since the video is reconstructed patch by patch, and each of the inversions are performed independently. iii) The *analytic* inversion expression results in efficient reconstruction of every patch. iv) Confidence of reconstruction on every video pixel is provided by the proposed method's probabilistic information *"error-bars"*.

Another contribution of this paper is the adaptive integration window. We utilize the reconstruction information of the proposed inversion method and learn a multiclass classifier to adapt T assuming fixed δ_t. Simulation results demonstrate that our method can adaptively change T subject to the motion of the objects in the scene while maintaining high-quality reconstruction performance. Future work lies in the implementation of adaptive integration window to the real CACTI system.

The GMM-based inversion algorithm developed in this paper can be straightforwardly extended to invert the hyperspectral datacubes from compressive measurements as discussed in [58]. Compared with the BPFA algorithm developed therein, the GMM-based inversion method enjoys fast reconstruction. Finally, this algorithm's low order of complexity lends itself to future higher-dimensional imagery; in particular, we seek to extend the proposed inversion method to reconstruct 4D spatio-spectral-temporal data volumes from 2D compressed measurements in the near future.

References

[1] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, February 2006.

[2] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[3] D. Reddy, A. Veeraraghavan, and R. Chellappa, "P2C2: Programmable pixel compressive camera for high speed imaging," *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 329–336, June 2011.

[4] Y. Hitomi, J. Gu, M. Gupta, T. Mitsunaga, and S. K. Nayar, "Video from a single coded exposure photograph using a learned over-complete dictionary," *IEEE International Conference on Computer Vision (ICCV)*, pp. 287–294, November 2011.

[5] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. Brady, "Coded aperture compressive temporal imaging," *Optics Express*, vol. 21, no. 9, pp. 10 526–10 545, 2013.

[6] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin, "Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 130–144, January 2012.

[7] Z. Xing, M. Zhou, A. Castrodad, G. Sapiro, and L. Carin, "Dictionary learning for noisy and incomplete hyperspectral images," *SIAM Journal on Imaging Sciences*, vol. 5, no. 1, pp. 33–56, Jan. 2012.

[8] A. C. Sankaranarayanan, C. Studer, and R. G. Baraniuk, "CS-MUVI: Video compressive sensing for spatial-multiplexing cameras," *IEEE International Conference on Computational Photography*, pp. 1–10, April 2012.

[9] Z. Liu, A. Y. Elezzabi, and H. V. Zhao, "Maximum frame rate video acquisition using adaptive compressed sensing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 11, pp. 1704–1718, November 2011.

[10] J. E. Fowler, S. Mun, and E. W. Tramel, "Block-based compressed sensing of images and video," *Foundations and Trends in Signal Processing*, vol. 4, no. 4, pp. 297–416, 2012.

[11] M. F. Duarte, M. A.Davenport, D. Takhar, a. S. T. J. N. Laska, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via com-

pressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, March 2008.

[12] A. Veeraraghavan, D. Reddy, and R. Raskar, "Coded strobing photography: Compressive sensing of high speed periodic videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 671–686, April 2011.

[13] M. B. Wakin, J. N. Laska, M. F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "Compressive imaging for video representation and coding," *Proceedings of the Picture Coding Symposium*, pp. 1–6, April 2006.

[14] J. Holloway, A. C. Sankaranarayanan, A. Veeraraghavan, and S. Tambe, "Flutter shutter video camera for compressive sensing of videos," *Intl. Conf. Computational Photography*, pp. 1–9, April 2012.

[15] S. K. Nayar, V. Branzoi, and T. E. Boult, "Programmable Imaging: Towards a Flexible Camera," *International Journal on Computer Vision*, pp. 7–22, Oct 2006.

[16] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: motion deblurring using fluttered shutter," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 795–804, 2006.

[17] A. Agrawal, M. Gupta, A. Veeraraghavan, and S. Narasimhan, "Optimal coded sampling for temporal super-resolution," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 599–606, June 2010.

[18] M. Gupta, A. Agrawal, A. Veeraraghavan, and S. G. Narasimhan, "Flexible voxels for motion-aware videography," *11th European Conference on Computer Vision, Part I*, pp. 100–114, September 2010.

[19] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. Brady, "Compressive sensing for video using a passive coding element," *Computational Optical Sensing and Imaging*, pp. 1–3, 2013.

[20] A. C. Sankaranarayanan, P. K. Turaga, R. G. Baraniuk, and R. Chellappa, "Compressive acquisition of dynamic scenes," *11th European Conference on Computer Vision, Part I*, pp. 129–142, September 2010.

[21] J. Y. Park and M. B. Wakin, "A multiscale framework for compressive sensing of video," *Proceedings of the Picture Coding Symposium*, pp. 1–4, May 2009.

[22] ——, "Multiscale algorithm for reconstructing videos from streaming compressive measurements," *Journal of Electronic Imaging*, vol. 22, no. 2, pp. 1–17, 2013.

[23] C. Liu, "Beyond pixels: Exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, May 2009.

[24] S. Mun and J. E. Fowler, "Residual reconstruction for block-based compressed sensing of video," *Data Compression Conference*, pp. 183–192, March 2011.

[25] D. Kittle, K. Choi, A. Wagadarikar, and D. J. Brady, "Multiframe image estimation for coded aperture snapshot spectral imagers," *Applied Optics*, vol. 49, no. 36, pp. 6824–6833, December 2010.

[26] H. Permuter, J. Francos, and I. H. Jermyn, "Gaussian mixture models of texture and colour for image database retrieval," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, pp. 569–72, April 2003.

[27] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 246–252, 1999.

[28] G. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2481–2499, May 2012.

[29] J. A. Guerrero-Colon, L. Mancera, and J. Portilla, "Image restoration using space-variant Gaussian scale mixtures in overcomplete pyramids," *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 27–41, January 2008.

[30] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin, "Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6140–6155, December 2010.

[31] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Transactions on Information Theory*, vol. 55, no. 11, pp. 5302–5316, November 2009.

[32] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 3042–3054, November 2010.

[33] J. Yang, X. Yuan, X. Liao, P. Llull, G. Sapiro, D. J. Brady, and L. Carin, "Gaussian Mixture Model for Video Compressive Sensing," *International Conference on Image Processing*, pp. 19–23, 2013.

[34] W. Carson, M. Chen, M. Rodrigues, R. Calderbank, and L. Carin, "Communications inspired projection design with application to compressive sensing," *SIAM Journal on Imaging Sciences*, vol. 5, no. 4, pp. 1185–1212, 2012.

[35] J. M. Duarte-Carvajalino, G. Yu, L. Carin, and G. Sapiro, "Task-driven adaptive statistical compressive sensing of Gaussian mixture models," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 585–600, February 2013.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[37] A. Corduneanu and C. M. Bishop, "Variational Bayesian model selection for mixture distributions," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2001, pp. 27–34.

[38] M. D. Hoffman, D. M. Blei, and F. Bach, "Online learning for latent dirichlet allocation," *Proceedings of the Neural Information Processing Systems*, pp. 856–864, 2010.

[39] L. Li, J. Silva, M. Zhou, and L. Carin, "Online bayesian dictionary learning for large datasets," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2157–2160, 2012.

[40] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, November 2006.

[41] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM review*, no. 51, pp. 34–81, 2007.

[42] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[43] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society, Series B*, vol. 68, pp. 49–67, 2006.

[44] J. Silva, M. Chen, Y. C. Eldar, G. Sapiro, and L. Carin, "Blind compressed sensing over a structured union of subspaces," Arxiv preprint arXiv:1103.2469, Tech. Rep., 2011.

[45] F. Renna, R. Calderbank, L. Carin, and M. Rodrigues, "Reconstruction of signals drawn from a gaussian mixture via noisy compressive measurements," *IEEE Transactions on Signal Processing*, vol. 62, no. 9, pp. 2265–2277, 2014.

[46] G. Yu and G. Sapiro, "Statistical compressed sensing of Gaussian mixture models," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 5842–5858, December 2011.

[47] C. Ordonez and P. Cereghini, "SQLEM: Fast clustering in SQL using the EM algorithm," *In ACM SIGMOD Conference*, pp. 559–570, 2000.

[48] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[49] C.-H. Hsieh and T.-P. Lin, "VLSI architecture for block-matching motion estimation algorithm," *IEEE Transactions on*

Circuits and Systems for Video Technology, vol. 2, no. 2, pp. 169–175, June 1992.

[50] M. Ezhilarasan and P. Thambidurai, "Simplified block matching algorithm for fast motion estimation in video compression," *Journal of Computer Science*, vol. 4, no. 4, pp. 282–289, 2008.

[51] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, March 2002.

[52] J. Yang and I. W. Tsang, "Hierarchical maximum margin learning for multi-class classification," *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pp. 753 – 760, 2011.

[53] X. Yuan, J. Yang, P. Llull, X. Liao, G. Sapiro, D. J. Brady, and L. Carin, "Adaptive temporal compressive sensing for video," *International Conference on Image Processing*, pp. 14–18, 2013.

[54] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[55] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error measurement to structural similarity," *IEEE Trans. Image Processing*, vol. 13, pp. 600–612, 2004.

[56] X. Liao, H. Li, and L. Carin, "Generalized alternating projection for weighted-$\ell_{2,1}$ minimization with applications to model-based compressive sensing," *to appear in SIAM Journal on Imaging Sciences*, 2014.

[57] J. Bioucas-Dias and M. Figueiredo, "A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 2992–3004, December 2007.

[58] A. Rajwade, D. Kittle, T.-H. Tsai, and L. Carin, "Coded hyperspectral imaging and blind compressive sensing," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 782–812, 2013.

[59] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, December 2007.