Journal of Network and Computer Applications 117 (2018) 72–85

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Two-layer compressive sensing based video encoding and decoding framework for WMSN

Yang Yang[a,b], Songtao Guo[a,*], Guiyan Liu[a], Yuanyuan Yang[a,c]

[a] National & Local Joint Engineering Laboratory of Intelligent Transmission and Control Technology (Chongqing), College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China

[b] College of Computer and Information Science, Southwest University, Chongqing 400715, China

[c] Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA

ARTICLE INFO

Keywords:
Compressive sensing
Video compression
Sparse basis
Dictionary learning
Wireless sensor networks

ABSTRACT

Acquiring massive data from wireless devices such as video sampling and transmission remains a challenge in the resource-restricted wireless multimedia sensor networks (WMSNs). To address the conflict between massive data processing and limited bandwidth or energy, we propose a two-layer compressive sensing based video encoding and decoding framework, which not only significantly reduces the amount of sampling data, but also transfers computation burden from distributed sensor nodes to powerful sink node. To successfully decode compressed data, at the first layer, we construct the first group sparse basis by exploiting the correlation between frames for each sensor node. At the second layer, we consider the spatial correlation between neighbor nodes. By employing dictionary learning, we obtain the second group sparse basis. The sink node employs both groups of sparse bases to perform l_1 norm minimization problem twice and recover original video for each node. Experiment results show that, our CS based framework has high compression efficiency due to our constructed sparse basis. Compared with existing sparse basis, our sparse basis provides better sparse representation.

1. Introduction

The rapid development of sensors, embedded computing and cheaper CMOS (Complementary Metal Oxide Semiconductor) fosters the emergence of a promising wireless multimedia sensor network (WMSN) (Akyildiz et al., 2007). Beside to gather, process, and transmit scalar data (i.e., temperature, pressure, humidity, or location) in WSN (Amjad et al., 2016; Rashid and Rehmani, 2016), WMSNs have been extended to support media rich content such as images, video and audio, and have extensively applied in surveillance and industrial process control. In these applications, WMSNs will produce huge volumes of multimedia data which require high transmission bandwidth and high computation ability. However, these requirements are in conflict with the limited computation resource and network bandwidth.

To deal with these challenges, decision fusion (Salvo Rossi et al., 2013, 2015) with fading channel has been proposed to reduce the communication power. Distributed MIMO (Ciuonzo et al., 2012) has been proposed to improve the throughput of channel. Qos/Qoe based routing (Usman et al., 2018; Ehsan and Hamdaoui, 2012), and multiple-path routing (Xu et al., 2012; Medjiah et al., 2012) have been proposed to minimize energy consumption. Cross-layer framework (Shah et al., 2012) has also been proposed to analyze the end-to-end delay or maximize the capacity of networks.

Among these methods, video encoding/decoding is regard as most direct and efficient way (Almalkawi et al., 2010) to reduce the volumes of raw video data. Since video encoding has the same effect as compression, we use these two words alternatively in this paper. Compression or encoding can be classified into many categories (ZainEldin et al., 2015), however, traditional compression or encoding methods are not practical for WMSNs because of their higher computation complexity on encoder.

Recently, as the most promising compression paradigm, compressive sensing (CS) has been studied widely in the field of image and video processing (Eldar and Kutyniok, 2012; Yang et al., 2014; Xie et al., 2009). Most of them focus on the reconstruction for original signals and the construction of measurement matrix, however, few work pays attention to the sparse representation of signals. Some novel sparse bases such as Karhunen-Loeve transform(KLT) (Liu et al., 2013) and Lifting-based invertible motion adaptive transform (LIMAT) (Park and Wakin, 2009), and some novel recovery algorithms implicitly including sparse basis (Yang et al., 2014), have been proposed to explore the correlation between adjacent frames. However, these sparse bases

* Corresponding author.
E-mail addresses: songtao_guo@163.com (S. Guo), yuanyuan.yang@stonybrook.edu (Y. Yang).

https://doi.org/10.1016/j.jnca.2018.05.018
Received 5 February 2018; Received in revised form 10 May 2018; Accepted 24 May 2018
Available online 29 May 2018
1084-8045/

are not straightforward and can cause a large number of iterations during recovery. Moreover, they are still time-consuming and poor efficiency. We will discuss them in details in Section 2.

On the other hand, beside temporal similarity on multimedia sensor node, spatial similarity is another consideration for reducing data volume where neighboring nodes sample similar scene. Some of previous works (Colonnese et al., 2013; Thirumalai and Frossard, 2013; Pan et al., 2015) studied the multi-view video model or coding. In (Sankaranarayanan et al., 2012; Trocan et al., 2014; Liu et al., 2017), CS based compression and recovery methods are employed to multi-view scenarios. However, these methods are based on original image or video. If images or videos are encoded in temporal domain, they no longer take effect. To the best of our knowledge, few work jointly considers both temporal correlation and spatial correlation.

In this paper, we propose a two-layer compressive sensing based video encoding/decoding (ComVideo) framework for WMSNs, which aims to reduce the complexity of encoding/decoding algorithm and the amount of video sampling data as well as the transmission cost, eventually prolonging the lifetime of WMSNs. At the first layer, each sensor node utilizes the CS to encode video frames individually. At the second layer, our framework enables aggregation nodes to further compress the assembled block data from neighboring nodes. For decoding, the sink node analyzes the temporal correlation between frames of a node to obtain the first group sparse basis. The sink node employs dictionary learning method to obtain the second group sparse basis and explores the spatial correlation of frames from neighboring nodes. The construction of two groups of sparse bases is performed on sink node instead of sensor nodes, which reduces the computation burden and time consumption of sensor nodes. Although training-and-applying scheme causes the time consumption on sink node, the second layer encoding/decoding is optional. When neighboring sensor nodes capture similar scene and the amount of traffics is a major concern in the network, our framework will activate the second layer encoding/decoding on aggregation node. In contrast, we can deactivate the second layer when the reconstruction time cost is sensitive.

Our work mainly focuses on the construction of sparse basis and our contributions are summarized as follows:

- Inspired by motion estimation in traditional video coding, we utilize neighboring blocks in key frame to construct the first group of sparse basis directly. Compared with fixed transform basis (e.g. Discrete cosine transform (DCT) or Discrete cosine transform (DWT)) and existing stat-of-the arts methods (Liu et al., 2013; Asif et al., 2013; Chen et al., 2011), our sparse basis provides better recovery quality, and its construction is more concise.
- By employing dictionary learning to explore spatial sparsity, we obtain the second group of sparse basis for sparse representation of similar video frames from different sensor nodes, which enables compressing these similar video further at aggregation nodes.
- Numerical results show that our two-layer framework can greatly reduce video data amount in WMSNs. Compared to DVC (Distributed video coding) or similar multiple-layer compression methods (Hirokawa et al., 2017), most of computations are transferred to sink node and the average lifetime of sensor nodes is prolonged.

The rest of the paper is organized as follows. Section 2 reviews related works about video encoding or compression method in WMSNs, especially for source coding based compression method and CS based compression method. Section 3 presents the architecture of two-layer encoding/decoding framework. In Section 4, we describe the first layer encoding/decoding and the construction of first group sparse basis. Section 5 gives the second layer encoding/decoding. We describe the process of the second group sparse basis obtained by dictionary learning. In Section 6, we evaluate the performance of our framework. Section 7 concludes this paper.

2. Related work

In this section, we firstly review the related works on encoding or compression in WMSNs, then introduce some related works on CS theory.

2.1. Source coding based encoding methods

As aforementioned in Section 1, compression and encoding are more efficient way to reduce the traffic amount in network (Akyildiz et al., 2007). Although traditional encoding methods such as JPEG, MPEG-4 and H.264 have been well investigated, they are not suitable for WMSNs because high computation burden at encoding node. It is not practical to execute complex algorithms on sensor nodes powered by batteries. Even if some previous works (Rein and Reisslein, 2011) have proposed the improved encoding methods, the results are still unsatisfactory. Moreover, other filter based encoding method or collaborative hybrid learning algorithm (Wang et al., 2008) also bring limited benefits.

DVC(Distributed video coding) as the most promising paradigm has been proposed in (Puri et al., 2006). It is based on DSC (Distributed Source Coding) (Xiong et al., 2004) and the main idea is traced from information theory established by Slepian-Wolf and Wyner-Ziv. DVC paradigm has the following advantages: (i) Each sensor independently encodes its output to a sink node for jointly decoding; (ii) Its many-to-one coding paradigm shifts the computation complexity at encoder side to the powerful decoder side. However, one of its disadvantages is that the feedback between decoder and encoder may cause extra delay. Moreover, the insufficiently encoded data from the encoder will result in decoding failure since the encoder has no knowledge of side information during encoding.

2.2. CS based encoding methods in WMSNs

In this subsection, we will discuss CS based video processing methods. In recent years, compressive sensing (Donoho, 2006) is an active topic in many research fields, particularly, in multimedia field for image processing such as deionising, recovery, super-resolution and compression (Metzler et al., 2016; Asif et al., 2013). However, the works of combining WMSNs with CS theory are not enough yet. The work in (Pudlewski et al., 2012) presents a general framework for WMSNs that jointly considers video encoding based on CS with transmission rate. However, how to employ CS to encoding/decoding videos is not provided. From the perspective of compressing single video, early works mainly exploit intra-frame correlation (i.e., through DCT, DWT, DFT transform basis) to compress each video frame. Since these transform bases cannot provide best sparsity, the recovery quality is not high and still needs to be improved. Intra-frame correlation ignores the similarity and consecutiveness between frames, thus the recent works mainly employ inter-frame correlation to recover encoded video frames. Karhunen-Loeve transform (KLT) (Liu et al., 2013) basis was proposed to explore correlation between adjacent frames, and an iterative updating method was applied to decoder. However the construction of KLT basis needs backward-direction and forward-direction iteration and singular value decomposition (SVD). Thus it has high time-consuming and computation burden.

In (Azghani et al., 2016), a multi-hypothesis (MH) reconstruction strategy was proposed for exploiting the sparsity of the video frames, where the cost function is minimized by using the Alternating Direction Method of Multipliers (ADMM). A motion-adaptive dynamical model was adopted in (Asif et al., 2013) to keep track of the interpolation operators. To recover encoded frames, this model iteratively updates motion estimation of video frames within adjacent frames. In (Liu and Pados, 2013), a Total-variation (TV) minimization scheme was applied to model gradient sparsity, which can achieve the same effect as sparse basis. In (Yang et al., 2014), Gaussian mixture model (GMM) was used

to model temporal correlation. The GMM-based inversion method employs online adaptive learning and parallel computation and the online-updated GMM method can yield good reconstruction results even with unrelated training videos. However, it is worth noting that most existing dictionary learning methods have high computational cost, in particular when it is integrated into the GMM-based method. However, as aforementioned, these sparse bases are not straightforward and effective. Most of these schemes require complicated computation or iterative operation. High computation burden makes these sparse bases not suitable for the resource-restricted WMSNs, even if the sink node is generally assumed to have powerful computation ability. In addition, these works just focus on single video compression in temporal domain.

From the perspective of compressing multiple similar videos from different sensor nodes, CS based encoding methods are employed to multi-view scenarios in (Thirumalai and Frossard, 2013; Trocan et al., 2014; Liu et al., 2017). In a network, beside temporal correlation between consecutive videos at multimedia sensor node, spatial correlation among neighboring nodes should also be considered for further reducing data size. In (Trocan et al., 2014), an iterative disparity estimation and compensation algorithm was proposed for the reconstruction of multiview images, where wiener filtering combine projected Landweber (PL) algorithm is iteratively applied in disparity estimation and compensation. Similarly, disparity and motion-compensated method was adopted in (Liu et al., 2017) with TV minimization algorithm to jointly reconstruct the multiview videos.

In (Sankaranarayanan et al., 2012), optical flow based compressing method was integrated into recovery algorithm to create a low-resolution video preview. Then the video preview problem was formulated into a convex optimization to recover the high-resolution video. The similar idea was adopted in (Ebrahim and Chia, 2016), where authors employed twice decoding and fusion strategy to recover multiview compressed videos. However, these methods work on original images or videos. If images or videos have been encoded in temporal domain, these methods no longer take effect. Moreover, few work jointly considers both temporal correlation and spatial correlation. In (Hirokawa et al., 2017), a two-layer compression scheme was provided, where the first layer employs CS to compress and the second layer uses H.264 intra coding and Wyner-Ziv coding to compress again. However, the scheme just uses the DCT transform as sparse basis of first layer. As for the second-layer compression, it treats the encoded data as an image to compress again. This implies that the second layer compression is totally different from our framework, which concerns the correlation of multiple neighboring nodes.

3. Framework of two-layers encoding and decoding

In this section, we first elaborate our two-layer framework, and then present the advantage of cluster topology and our concerns in this paper.

3.1. Our two-layer encoding and decoding framework

In WMSNs, sensor nodes are organized as a cluster as depicted in Fig. 1, where there are three types of nodes: common nodes, aggregation nodes and sink node. The first two types of nodes are low-cost and low-power consumption nodes. They perform simple encoding operations. Sink node executes video decoding and recovery. Since the amount of video data is far more than that of 1-D scalar data, our goal is to maximize compression ratio while ensuring recovery quality. Thus we apply CS theory twice in our framework, which is called two-layer framework. The first layer encoding is based on time domain and executed by each sensor node. The second layer encoding is based on spatial domain and performed by aggregation node.

Fig. 2 illustrates the encoding process of our ComVideo framework. For the sake of facilitating explanation, we take one cluster as observation object without loss of generality. Suppose that sensor nodes in

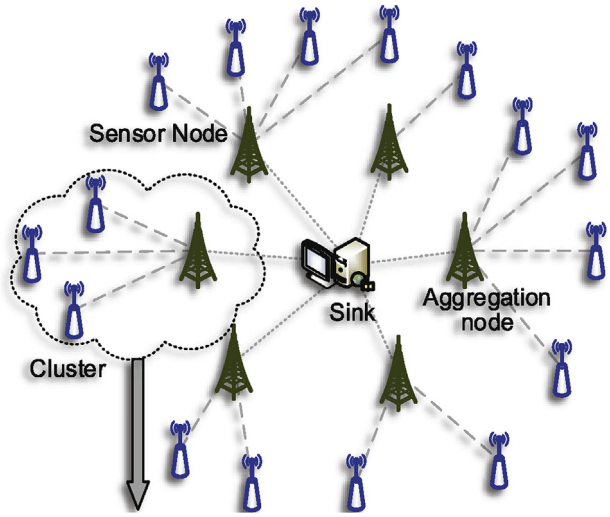


Fig. 1. The topology of our framework, where both sensor nodes and aggregation nodes are encoders, and sink node is decoder.

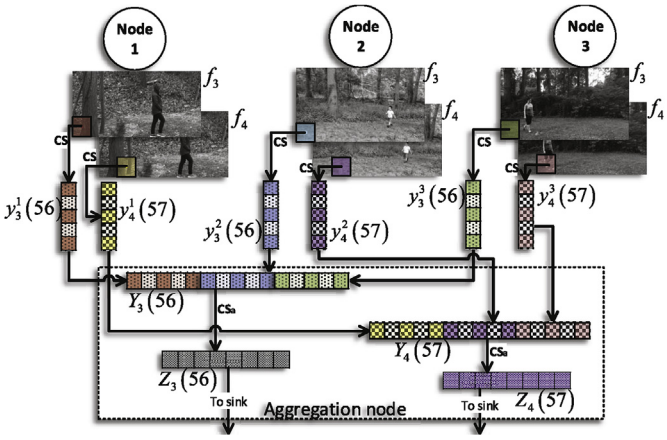


Fig. 2. Illustration of two-layer encoding process. Each node encodes its data at temporal domain layer while aggregation node encodes the data of multiple nodes at spatial domain layer.

a cluster are denoted by the set $\mathcal{N} = \{s_1, s_2, ..., s_n\}$. After a period of time t, each sensor node produces a video clip, which consists of a group of correlated and consecutive frames, denoted by $\mathcal{F} = \{f_1, f_2, ..., f_t\}$. Similar to DVC, we divide the correlated frames into two types: Key frames f_r and CS frames f_{cs}. Key frames are used as reference for CS frames. Specifically, key frame is the frame with low compression ratio while CS frame is the frame with high compression ratio. After being encoded by CS, key frame is recovered by $\|\cdot\|_1$ minimization method, however, the recovery of CS frame depends on temporal sparse basis being constructed by referring neighboring blocks in key frame.

Each CS frame f_i is divided into K non-overlapping image blocks. Each block has a sequence number in its frame. Hence, each video is divided into many image blocks. These blocks have three attributes: node identity, frame identity, and block sequence number in one specific frame. We use $b_i^j(k)$, $i \in \mathcal{F}$, $j \in \mathcal{N}$, $k \in \{1, 2, ..., K\}$ to denote an image block. Table 1 gives the list of notations which will be used later.

At the temporal domain layer of our framework, each sensor node vectorizes its image blocks, and encodes these vectors by multiplying block measurement matrix ϕ_b, then we get corresponding encoded vector $y_i^j(k)$.

$$y_i^j(k) = \phi_b b_i^j(k) \tag{1}$$

For example, in Fig. 2, six image blocks come from two consecutive

Table 1
List of notations.

Notation	Definition
s_i	sensor node i within cluster $\mathcal{N}$
f_i	video frame i in video clip $\mathcal{F}$
$b_i^j(k)$	the kth image block which belongs to the ith frame of jth sensor node
$\psi_i^j(k)$	sparse basis corresponding to image block $b_i^j(k)$
$\theta_i^j(k)$	sparse vector corresponding to $b_i^j(k)$
$y_i^j(k)$	encoded vector corresponding to $b_i^j(k)$
ϕ_b	measurement matrix applied to each blocks $b_i^j(k)$
$Y_i(k)$	assembled vector which contains $y_i^j(k)$ from different nodes, but has the same frame i and block sequence number k
ϕ_s	measurement matrix applied to assembled vector $Y_i(k)$
$Z_i(k)$	encoded vector corresponding to $Y_i(k)$
$D_i(k)$	sparse dictionary corresponding to $Y_i(k)$
$S_i(k)$	sparse vector corresponding to $Y_i(k)$

frames (f_3, f_4) of three different nodes (s_1, s_2, s_3). Through formula (1), these blocks are encoded to vectors $y_{f_3}^{s_1}(56)$, $y_{f_4}^{s_1}(57)$, $y_{f_3}^{s_2}(56)$, $y_{f_4}^{s_2}(57)$, $y_{f_3}^{s_3}(56)$, $y_{f_4}^{s_3}(57)$. Sensor nodes send the encoded vector $y_i^j(k)$ to aggregation node. After that, the temporal domain layer encoding is finished. At the spatial domain layer, aggregation node assembles these vectors which come from different nodes but have the same frame identity and block sequence number (e.g. $y_{f_3}^{s_1}(56)$, $y_{f_3}^{s_2}(56)$, $y_{f_3}^{s_3}(56)$), into one vector $Y_i(k)$ (e.g. $Y_{f_3}(56)$). Then we apply CS theory again, and get encoded vector $Z_i(k)$:

$$Z_i(k) = \phi_s Y_i(k) \tag{2}$$

where ϕ_s is called spatial measurement matrix. We choose both ϕ_b and ϕ_s as random Gaussian matrix according to (Donoho, 2006). Subsequently, aggregation node quantizes the vector $Z_i(k)$ and sends quantization result to sink node. Since $Z_i(k)$ contains the information of k-th block in i-th frame from all sensor nodes, after receiving the vector $Z_i = \bigcup_{k=1}^{K} Z_i(k)$, sink node gets the entire frame of all sensor nodes in the cluster. Finally, the sink node accumulates all the frames $Z = \bigcup_{i=f_1}^{f_t} Z_i$. By the similar way, the sink node can receive all video frames in one cluster.

At the sink node, two-layer decoding algorithm is performed inversely. At the spatial domain layer, $Y_i(k)$ is retrieved from $Z_i(k)$ by solving the following problem:

$$\min_{S_i(k)} \|S_i(k)\|_1 \quad s.t. \;\; Z_i(k) = \phi_s D_i(k) S_i(k) \tag{3}$$

$$Y_i(k) = D_i(k) S_i(k) \tag{4}$$

where $D_i(k)$ is sparse dictionary of $Y_i(k)$, and $S_i(k)$ is the corresponding sparse vector. At the temporal domain layer, the restored $Y_i(k)$ is divided into several $y_i^j(k)$, according to different nodes. Then block $b_i^j(k)$ is recovered from encoding vector $y_i^j(k)$ by solving the following problem:

$$\min_{\theta_i^j(k)} \left\| \theta_i^j(k) \right\|_1 \quad s.t. \;\; y_i^j(k) = \phi_b \psi_i^j(k) \theta_i^j(k) \tag{5}$$

$$b_i^j(k) = \psi_i^j(k) \theta_i^j(k) \tag{6}$$

where $\psi_i^j(k)$ is sparse basis of block $b_i^j(k)$, and $\theta_i^j(k)$ is the corresponding sparse vector. The blocks with the same frame identity and node identity (e.g. $b_{f_3}^{s_1}(1)$, $b_{f_3}^{s_1}(2)$, ...,$y_{f_3}^{s_1}(K)$) are reorganized into one frame (e.g. $f_3^{s_1}$) according to block sequence number k. Similarly, a group of frames belonging to one node (e.g. $f_1^{s_1}, f_2^{s_1}$, ...,$f_t^{s_1}$,) are reorganized into one video by frame identity i. Accordingly, we finish the video recovery of one node.

3.2. Feature of our framework

It is not difficult to observe that our framework has several advantages for WMSNs. Firstly, the encoding in our framework is simple and easy to be implemented. After matrix multiplication in (1) and (2), the unprocessed data are encoded and compressed simultaneously. The sensor nodes, as encoders, just perform simple matrix multiplication. Most computation burdens i.e., sparse basis construction and solving l_1 minimization problem, are transferred to sink node. Secondly, our framework enables aggregation nodes to compress video data from different nodes, which can further reduce network traffic and transmission cost. In addition, our framework is flexible. To elaborate, because the spatial domain layer is auxiliary, we can activate it when we need to improve overall compression ratio. In this case the aggregation nodes will perform two functions, i.e., compression and relaying data. In contrast, we can deactivate the spatial domain layer when we pay attention to reconstruction time cost. In this case, the aggregation nodes only relay data to sink node without compression.

According to (Donoho, 2006), the following three preconditions need to be satisfied when applying CS: 1) Original signals can be sparsely represented, that is, the sparse basis Ψ can be found. 2) Measurement matrix Φ needs to be carefully chosen to make sure matrix $\Phi\Psi$ satisfying the RIP requirement (Donoho, 2006). 3) Sparse solution algorithms should be given to solve the optimization problem (3) and (5). In this paper, we aim to construct appropriate sparse basis Ψ for two layers. Recovering $Y_i(k)$ and $b_i^j(k)$ involves solving l_1 norm minimization problems, i.e., the optimization problems (3) and (5). To this end, we need to provide sparse basis $D_i(k)$ and $\psi_i^j(k)$. The better sparse representation we can get from sparse basis, the higher compression ratio we can achieve or the better recovery quality we can get. Once we get sparse basis $D_i(k)$ and $\psi_i^j(k)$, (3) and (5) can be solved by BP (Basis Pursuit) algorithm (Candes and Tao, 2005), which is a classical algorithm for solving l_1 norm minimization problem. We utilize some universal measurement matrices Φ given in (Donoho, 2006) to satisfy RIP requirement. This means that we do not need to design the sparse solution algorithm in this paper. Therefore, we mainly pay attention to the construction of sparse bases $\psi_i^j(k)$ and $D_i(k)$. In the following, we will present the construction of the two sparse bases in Section 4 and 5, respectively.

4. Temporal domain layer decoding and construction of sparse basis

At the temporal domain layer, each node encodes its video frames in the time domain and sink node executes the corresponding decoding. We design the group of pictures (GOP) structure with two key frames and five CS frames. The first and second CS frames are reconstructed by the first key frame. The fourth and fifth CS frames are reconstructed by the last key frame. The third CS frame is constructed by two key frames. Fig. 3 depicts the process of temporal domain layer encoding/decoding.

4.1. Encoding and decoding of key frame

The key frame provides motion estimation (ME) references for CS frames. The recovery of CS frame relies on the recovered key frame. Thus the quality of encoded key frame is very important. Instead of using fixed sparse basis, we reconstruct key frame by TV minimization method (Liu and Pados, 2013). The method can find the “smoothest” solution within gradient domain, and provides better sparsity than DCT or DWT. At the sensor nodes, the key frame f_r is stacked into 1-D vector, and multiplied by measurement matrix Φ_k. Then key frame f_r is encoded into Y_r, i.e., $Y_r = \Phi_r f_r$, $\Phi_r \in \mathbb{R}^{M_r \times N_r}$, where N_r corresponds to total pixels in the key frame. The sample ratio of key frames, here called measurement ratio, is defined by $\gamma_r = M_r/N_r$. The compression rate is defined by $\mu_r = 1 - \gamma_r$. Suppose f_r^{ij} denotes current pixel in the frame f_r. We can define

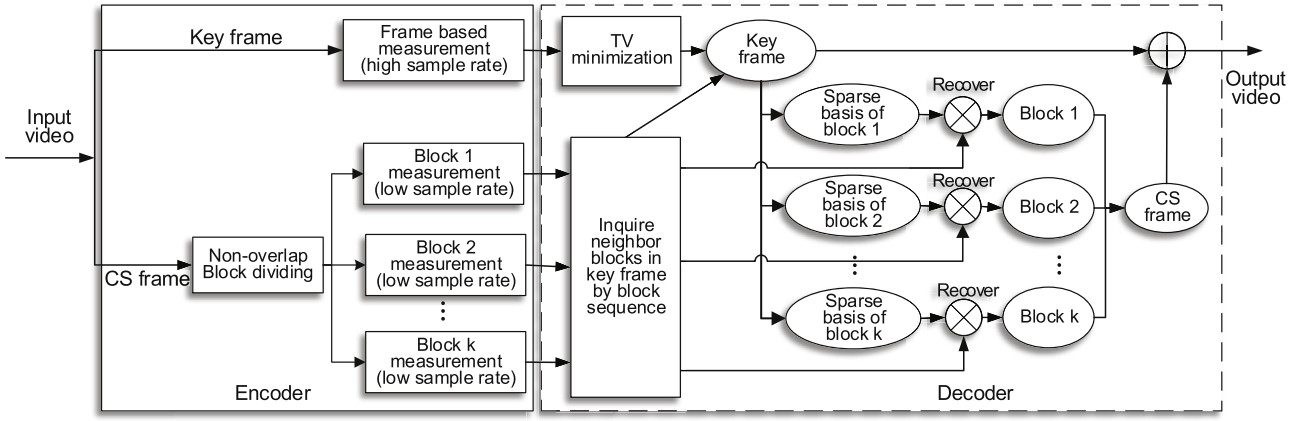


Fig. 3. The architecture of temporal domain layer encoding/decoding. Image blocks in each CS frame are encoded at sensor node. They are decoded and reassembled at sink node. Decoding algorithm uses sparse basis which is constructed by Key frame.

$$TV(f_r) = \Sigma_{i,j}|f_r^{i+1,j} - f_r^{i,j}| + |f_r^{i,j+1} - f_r^{i,j}| \tag{7}$$

the recovery problem of key frame can be stated as

$$\min TV(f_r) \quad s.t. \quad Y_r = \Phi_r f_r \qquad or \tag{8}$$

$$f_r = \arg\min_{f_r} \|Y_r - \Phi_r f_r\|_2 + \lambda\ TV(f_r) \tag{9}$$

Φ_r is an orthogonalized random Gaussian matrix. After obtaining f_r, we reshape it into 2-D image data and finish the recovery of key frame.

4.2. Encoding and decoding of CS frame

In this subsection, we will present recovery algorithm about CS frame and explore the construction of sparse basis $\psi_i^j(k)$ which is used for decoding $b_i^j(k)$ in (5) and (6). As illustrated in Fig. 3, each CS frame is divided into k non-overlapping blocks $b_i^j(k)$, and these blocks are encoded by $\phi_b \in \mathbb{R}^{m_b \times n_b}$ according to (1). Encoded vector $y_i^j(k)$ is sent to sink node. Sink node provides sparse basis $\psi_i^j(k)$ and matrix ϕ_b to BP algorithm. BP algorithm solves problem (5) and gets sparse vector $\theta_i^j(k)$, and $b_i^j(k)$ is recovered by $\psi_i^j(k)$ and $\theta_i^j(k)$ according to (6). We define measurement ratio γ_b of blocks as

$$\gamma_b = m_b/n_b \tag{10}$$

where n_b is size of a block. Suppose every block takes the same block measurement matrix ϕ_b, then the measurement ratio of frames and videos is also γ_b. In this paper, we unify ϕ_b of all sensor nodes.

To construct sparse basis $\psi_i^j(k)$, we exploit the temporal correlation between frames. Inspired by traditional encoding algorithm, our method takes advantage of similar block motion estimation. However, our block prediction is executed at sink node, which has powerful computation capability. We utilize neighboring blocks in key frame to construct $\psi_i^j(k)$. Figs. 4 and 5 illustrate the procedure. In Fig. 4, $b_i^j(k)$ is current block to be reconstructed, which is denoted as red in current CS frame. In key frame, we find the same position of $b_i^j(k)$ by parameter k. Taking that position as center and R_s as searching radius, we define a search area with size of $(2R_s + \sqrt{n_b})^2$. Then we use a mask block b_m, the size of which equals to $b_i^j(k)$, to slide in key frame, from left-up corner to right-bottom corner in search area. Sliding step size is one pixel. On each sliding step, we vectorize the block b_m, and add this vector to sparse basis $\psi_b^i(k)$ column by column. When sliding is finished, we obtain a complete sparse basis $\psi_b^i(k)$.

After parse basis $\psi_b^i(k)$ is obtained, sink node begins to solve problem (5). l_1 minimization algorithm forces most of items in the sparse vector $\theta_i^j(k)$ to be zeros. The sparse vector $\theta_i^j(k)$ is a similarity vector, which indicates the similarity coefficients between block $b_i^j(k)$ and neighboring blocks in key frame. As shown in Fig. 5, the maximal value in $\theta_i^j(k)$ is the third item. It means the block of the third column of $\psi_i^j(k)$ has similarity coefficient of 0.88 with current block $b_i^j(k)$. Finally, $b_i^j(k)$ is reconstructed through Eq. (6). We regard $b_i^j(k)$ as liner weighted sum of $\psi_i^j(k)$. Algorithm 1 summarizes the recovery process of one CS frame.

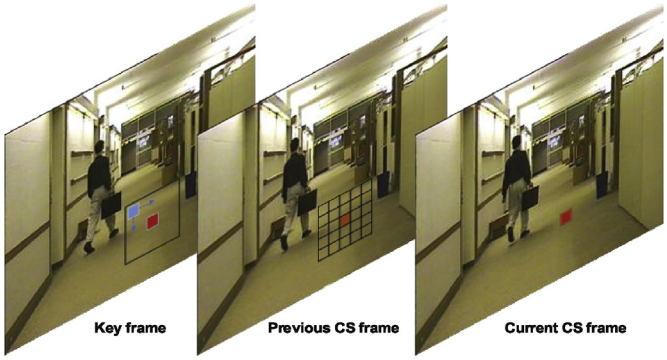


Fig. 4. The construction of sparse basis. Red block is target block, and blue block is mask block. Mask block slides in searching area. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

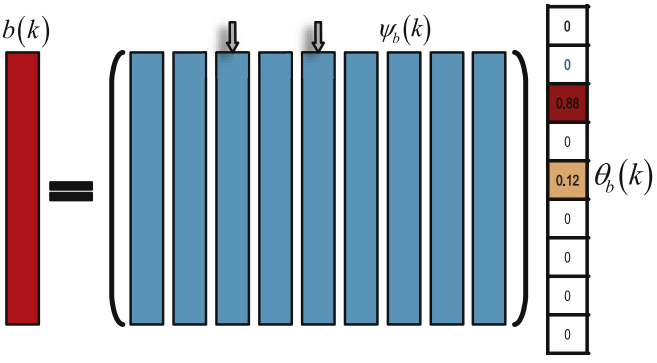


Fig. 5. Sparse basis $\psi_i^j(k)$ is constructed from key frame. Sparse vector $\theta_i^j(k)$ contains the information of position of similar blocks and its similarity coefficients.

Algorithm 1
CS frame recovery algorithm for time domain compression.

Input:
all encoded vectors $\bigcup_{k=1}^{K} y_i^j(k)$ of CS frame f_i in node n_j;
block size n_b, searching radius R_s, measurement matrix ϕ_b,
recovered key frame f_r;
Output
recovered CS frame f_i of one sensor node;
1: **for** each vector $y_i^j(k)$ **do**
2: initialize sparse basis $\psi_i^j(k) = \varnothing$; create mask block b_m with the size of n_b; locate corresponding block b_r in f_r by sequence k;
3: take b_r as center, expand search area with size of $(2R_s + \sqrt{n_b})^2$; slide b_m in search area by one pixel for each step;
4: vectorize b_m on each step, and add it into $\psi_i^j(k)$ as column, $\psi_i^j(k) = \psi_i^j(k) \bigcup b_m$;
5: solve $\arg\min_{\theta_i^j(k)} \left\| \theta_i^j(k) \right\|_1 \ s.t. \ y_i^j(k) = \phi_b \psi_i^j(k) \theta_i^j(k)$;
6: recover $b_i^j(k)$ by $b_i^j(k) = \psi_i^j(k)\theta_i^j(k)$; reshape vector $b_i^j(k)$ in the form of block;
7: **end for**
8: reorganize all recovered vectors $\bigcup_{k=1}^{K} b_i^j(k)$ into one frame;

In Algorithm 1, we use BP algorithm to solve l_1 minimization problem (6). Since the complexity of BP algorithm is $O(N^3)$, where N is size of sparse vector $\theta_i^j(k)$. The complexity of Algorithm 1 is $O((2R_s + \sqrt{n_b})^3)$. If we set searching radius as two times of side length of block, i,e, $R_s = 2\sqrt{n_b}$, then the complexity of Algorithm 1 is $O(n_b^{3/2})$. As for key frame, it is encoded based on entire frame. Supposing that each frame has P pixels, we get the complexity of recovering key frame is $O(P^3)$ according to TV minimization recovery algorithm. For comparing frame based strategy with block based strategy, we suppose that each frame can be divided into K blocks, i.e., $P = Kn_b$. The recovery complexity of key frame is $O(P^3) \approx O(n_b^3)$, which is obviously larger than the complexity of block based strategy.

5. Spatial domain layer decoding and sparse dictionary learning

In this section, we elaborate the process of spatial domain layer decoding and the sparse dictionary learning. As aforementioned, the spatial domain layer encoding/decoding is optional. When neighboring sensor nodes capture similar scene and traffic amount is a major concern in the network, our framework will activate the spatial domain layer encoding/decoding on aggregation node. Aggregation node assembles the encoded data from different sensor nodes to a new vector, and applies CS to encode the assembled vector again. To decode the assembled vector successfully, we need to find the corresponding sparse basis. To achieve the goal, in this paper, we propose a training method based on dictionary learning (Tosic and Frossard, 2011).

5.1. Motivation of using sparse dictionary learning

Different from 1-D data in traditional WSNs, spatial correlation and compression in WMSNs are much more intractable. Most of CS based works about WSNs assume that the data being sampled from networks are sparse in DCT or DWT transform basis. However, this assumption is unsuitable for 2-D data in WMSNs. The idea of the spatial domain layer encoding comes from the existing "multi-view reconstruction" works in image or video research field (Wu and Chen, 2007; Thirumalai and Frossard, 2013; Trocan et al., 2014; Liu et al., 2017). In (Wu and Chen, 2007), maximal block matching about multi-view video has been discussed, however, complex geometry projection algorithm is a heavy burden for sensor nodes and is not practical for WMSNs. Although the works (Thirumalai and Frossard, 2013; Trocan et al., 2014; Liu et al., 2017) have investigated the CS based or DSC based multi-view video, these methods only work on original frames, and are not suitable for the encoded data since our data of the spatial domain layer have been encoded.

In general, a certain blocks from different sensors with camera are not exactly same, however, they may be similar, or at least we can find the similar block around the specific position. The similarity provides the chance to sparsely represent the assembled vectors. After temporal domain layer encoding, we get the encoded vectors of image block, and we assemble these vectors from different nodes according to blocks position and frames. In the process, we exploit the correlation in the time domain but ignore the spatial correlation between nodes. In fact, we can further compress these similar frames from different nodes by using the spatial correlation. This is because by CS theory, any type of data can be compressed and recovered without considering the data context with slight deviation only if the data can be sparsely represented in a transform domain. Thus this motivates us to utilize the spatial domain layer compressive sensing to sparsely represent the assembled vectors.

Inspired by machine learning principle, where the parameters of a function can be obtained by learning from data, we find that using dictionary learning in the spatial domain layer compressive sensing is an effective way to sparsely represent the data used for training. However the key of our spatial domain layer decoding is to find appropriate sparse dictionary $D_i(k)$ for assembled vector $Y_i(k)$. Fortunately, sparse dictionary learning (Tosic and Frossard, 2011) makes the construction of $D_i(k)$ possible. Dictionary learning aims to find a sparse representation of the given data. The basic elements in sparse dictionary are called *atoms*. Compared to traditional transform basis, the atoms in the dictionary are not required to be orthogonal. K-SVD (Elad and Aharon, 2006) and MOD (Method of Optimal Directions) (Tosic and Frossard, 2011) are two major algorithms for training sparse dictionary. The key of dictionary learning is that the dictionary has to be trained from the data. In the following, we will describe how to train our sparse dictionary.

5.2. Sparse dictionary training and testing

Before activating the spatial domain layer encoding, the sink node employs the temporal domain layer decoding and get a portion of recovered video frames. The sink node uses these recovered frames, which are called training frame set $\mathscr{F}_{\text{tr}}$, to train sparse dictionary $D_i(k)$. Our training and recovery process includes three stages. At the first stage, our framework just activates the temporal domain layer encoding. Recovered blocks in training frames are assembled into a large vector $B_i(k) = (b_i^1(k), b_i^2(k), ..., b_i^n(k))^T$. These blocks come from different nodes with the same frame identity i and the same block sequence number k. The sink node puts these assembled vectors $B_i(k)$ with the same block sequence number k, into block training set $\mathscr{T}(k)$. After a period of time, $\mathscr{T}(k)$ accumulates all vectors $B_i(k)$ from training set $\mathscr{F}_{\text{tr}}$, i.e., $\mathscr{T}(k) = \{B_1(k), B_2(k), ..., B_q(k)\}$, where q denotes the number of frames in $\mathscr{F}_{\text{tr}}$. Furthermore, we can express $\mathscr{T}(k)$ in the form of matrix as

$$\mathscr{T}(k) = \begin{pmatrix} b_1^1(k) & b_2^1(k) & b_3^1(k) & \cdots & b_q^1(k) \\ b_1^2(k) & b_2^2(k) & b_3^2(k) & \cdots & b_q^2(k) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ b_1^n(k) & b_2^n(k) & b_3^n(k) & \cdots & b_q^n(k) \end{pmatrix}_{p \times q}$$

where $\mathscr{T}(k) \in \mathbb{R}^{p \times q}$ and $p = n \times n_b$ denotes the size of assembled vectors $B_i(k)$. $\mathscr{T}(k)$ represents specific block k from all nodes and all frames in $\mathscr{F}_{\text{tr}}$. The row of $\mathscr{T}(k)$ denotes all blocks in training frames of sensor node. The column of $\mathscr{T}(k)$ denotes all blocks in specific frame from different sensor nodes.

At the second stage, the spatial domain layer encoding is activated on aggregation node and $Z_i(k)$ is sent to sink node. We call these twice

encoded frames as testing frame set $\mathscr{F}_{ts}$. At the same time, sink node applies the improved K-SVD to train $\mathscr{T}(k)$. After training is finished, we can get the trained sparse dictionary $D_b(k)$ of vector $B_*(k)$.

$$B_*(k) = D_b(k)S_*(k) \tag{11}$$

where $*$ means any frames in $\mathscr{F}_{ts}$ and $S_*(k)$ is the corresponding sparse vector. However, our objective is to find a sparse dictionary of vector $Y_*(k)$, not only $B_*(k)$. According to (1), we rewrite assembled vector $Y_*(k)$ in the form of matrix by

$$Y_*(k) = \begin{bmatrix} y_*^1(k) \\ y_*^2(k) \\ \vdots \\ y_*^n(k) \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_b^1(k) & 0 & 0 & \cdots & 0 \\ 0 & \phi_b^2(k) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & \phi_b^n(k) \end{bmatrix}}_{\Phi_b(k)} \underbrace{\begin{bmatrix} b_*^1(k) \\ b_*^2(k) \\ \vdots \\ b_*^n(k) \end{bmatrix}}_{B_*(k)} \tag{12}$$

Combing with (11), we have

$$Y_*(k) = \Phi_b(k)D_b(k)S_*(k) \tag{13}$$

We get the sparse dictionary of assembled vector $Y_*(k)$ in testing frames as follows:

$$D_*(k) = \Phi_b(k)D_b(k) \tag{14}$$

At the third stage, after getting sparse dictionary of $D_*(k)$, the sink node decodes $Z_*(k)$ by solving (3), and recovers $Y_*(k)$ by (4). Here we have finished the spatial domain layer decoding. The spatial domain layer decoding is summarized as Algorithm 2.

Algorithm 2
Dictionary learning and spatial domain layer decoding.

Input:
encoded vectors $Z_i(k)$ from aggregation node; measurement matrix ϕ_b, ϕ_s;
recovered training frame in $\mathscr{F}_{\text{tr}}$;
Output:
vector $y_*^1(k)$, $y_*^2(k)$, ...,$y_*^n(k)$;
1: **for** each frame i in $\mathscr{F}_{\text{tr}}$ **do**
2: extract blocks $b_i^1(k)$, ...,$b_i^n(k)$ from different nodes;
3: assemble these blocks into vector $B_i(k)$, and insert $B_i(k)$ into matrix $\mathscr{T}(k)$;
4: **end for**
5: apply the improved K-SVD to train $\mathscr{T}(k)$ and get sparse dictionary $D_b(k)$ for block k;
6: construct diagonal matrix $\Phi_b(k)$ by using measurement matrixes $\phi_b^j(k)$ from different nodes;
7: construct sparse dictionary $D_*(k)$ for assembled vector $Y_*(k)$ according to (14);
8: **for** each encoded data $Z_i(k)$ in testing frame set $\mathscr{F}_{ts}$ **do**
9: solve (3) and get $S_*(k)$ and recover $Y_*(k)$ through (4)
10: break $Y_*(k)$ into several vectors $y_*^1(k)$, …, $y_*^n(k)$;
11: **end for**

After $Y_*(k)$ is divided into n vectors $(y_*^1(k), y_*^2(k),...,y_*^n(k))$ according to different nodes, sink node performs temporal domain layer decoding, as described in Section 4.2. After two-layer decoding is finished, we get recovery block k of frame f_*. Finally, sink node recovers all the frames in testing frames set. Combining training set with testing set, the sink node recovers entire video of all sensor nodes.

It is worth noting that, instead of using encoded vector $Y_*(k)$ to train $D_*(k)$ directly, we utilize the assembled $B_i(k)$ to train sparse dictionary $D_b(k)$, and obtain $D_*(k)$ by (14) indirectly. However, our method can provide better sparsity than direct training method, which will be illustrated in Section 6. In addition, we take two step recovery procedure because one step recovery procedure just exploit the sparsity of temporal domain layer as follows,

$$Z_*(k) = \Phi_s(k)\Phi_b(k)\Psi_b(k)\Theta_*(k). \tag{15}$$

We substitute $\Phi_s(k)\Psi_b(k)$ with a new matrix $\hat{\Psi_b}(k)$, then two step recovery procedure (15) is equivalent to temporal domain layer decoding. In fact, the sparsity of spatial domain layer is based on dictionary learning, thus the sparse vector $S_*(k)$ is different from $\Theta_*(k)$.

6. Performance evaluation

In this section, we will evaluate decoding performance of our framework. We execute our two layer encoding/decoding algorithm on a workstation with core I5-7500T processor, 16 GB memory, SSD storage. Since our framework contains two-layer encoding and decoding, we will also evaluate the performance of each layer and final recovery quality. We will compare our framework with other compression methods, like classic DVC scheme (Zhuo et al., 2008), CS based work which uses the sparse basis KLT (Liu et al., 2013), and the motion compensation related schemes (Trocan et al., 2014; Mun and Fowler, 2011; Liu et al., 2017). For the evaluation of the temporal domain layer encoding and decoding, we take four groups of videos for testing: two groups are outdoor surveillance video filmed by two sensors node in WMSNs, and other two groups come from public video library. For the spatial domain layer encoding and decoding, we take two groups of videos for comparison. One group is from public testing video “ballet” (Zitnick et al., 2004), and the other group comes from videos captured by our sensor cluster, which contains three sensor nodes and one aggregation node.

In our evaluation, all the related CS algorithms take random Gaussian matrix as measurement matrix (Donoho, 2006). Each element of ϕ_b and ϕ_s follows $\phi \sim N(0, 1)$. We set the block size is 8×8 and let $n_b = 64$ and search radius $R_s = 2\sqrt{n_b}$.

6.1. Evaluation for temporal domain layer encoding/decoding

In this subsection, we will evaluate encoding efficiency and recovery quality of CS frames based on our sparse basis. Fig. 6 depicts the visual recovery quality of two groups of recovered frames with different measurement ratios. The first group is based on public video “foreman”, and the second group is based on our captured video. Both of them have the 512×512 resolution. Fig. 6(a) and (b) show the adjacent key frame and original CS frame. The difference between key frame and CS frame of first group is the facing angle of foreman. The difference of second group frames is the right leg of Pedestrian. Fig. 6(c) and (f) show the recovery quality under different measurement ratios γ_b. We observe that the recovered frame is still recognizable even if γ_b is reduced from 50% to 6.25%, which means that the compression ratio reaches 93.75%. In Fig. 6(c) with $\gamma_b = 0.5$, we can hardly observe the difference between the recovered frame and the original frame. When $\gamma_b = 0.25$, some residuals can be found near the right leg of pedestrian by amplifying Fig. 6(d). When $\gamma_b = 0.125$, we find that the entire frame has a little blurring in Fig. 6(e), especially the moving parts around the pedestrian. We can observe from Fig. 6(f) that the entire frame has obvious mosaic effect and poor recovery quality and the peak signal-to-noise ratio (PSNR) is reduced to around 18.0 dB.

Furthermore, we compare our sparse basis with regular fixed transform basis which is used in the most of CS based works, KLT basis (Liu et al., 2013), multiple hypothesis (MH) (Azghani et al., 2016), and Gaussian mixture model(GMM) (Yang et al., 2014) based method. Fig. 7 shows the recovery results at the measurement ratio $\gamma_b = 0.25$. Obviously, our sparse basis and the ones in (Liu et al., 2013; Azghani et al., 2016; Yang et al., 2014) are superior to fixed transform basis. The recovery result of fixed transform basis is only acceptable when the measurement ratio is greater than 0.4. Compared with KLT, MH and GMM, our sparse basis provides the approximation result. Although our recovery quality is not best (Fig. 8(a)), we can find from Fig. 8(b) that

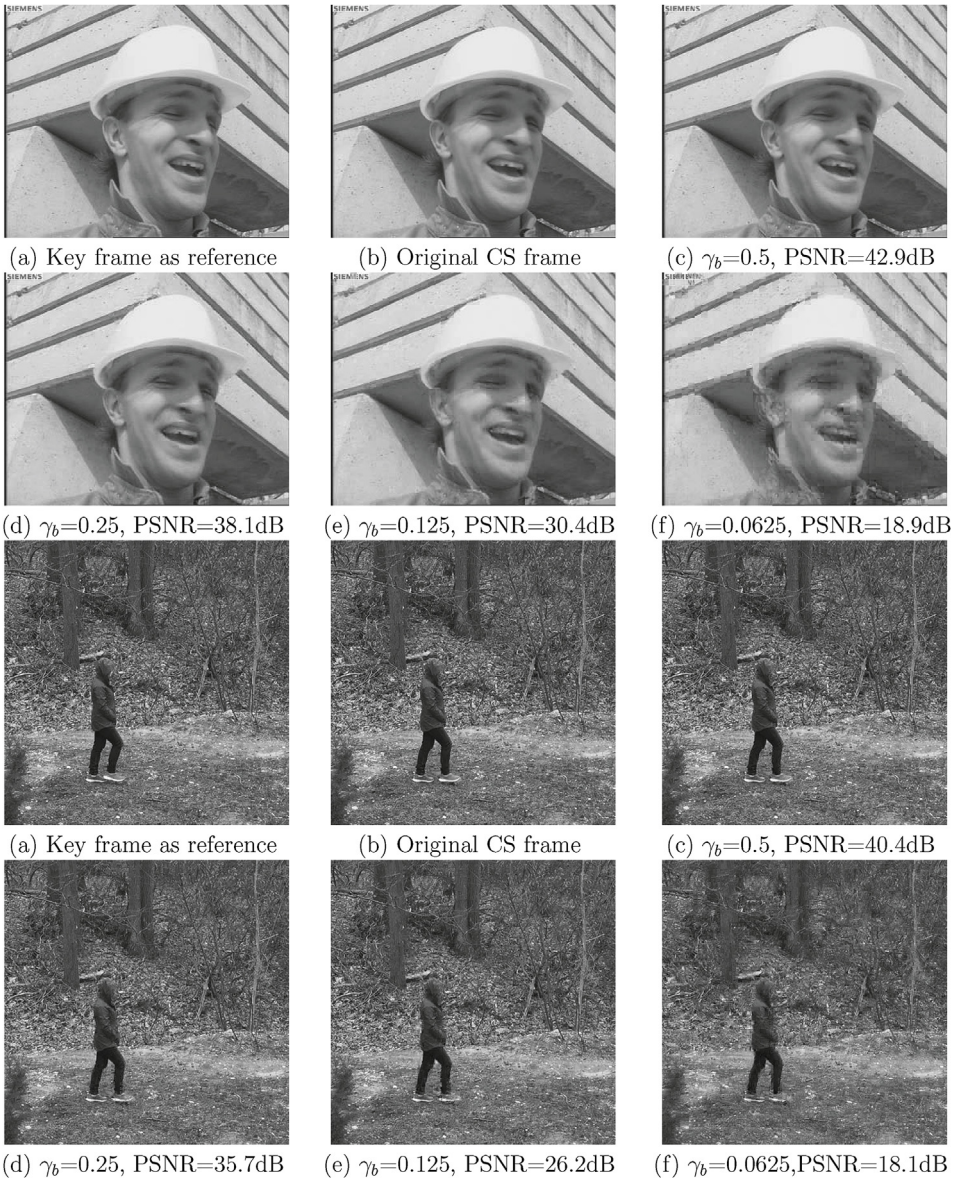


Fig. 6. CS recovery quality with different measurement ratios based on our sparse basis.

our sparse basis has lower time consumption during recovery procedure. This benefits from the concise and efficiency construction of our sparse basis.

In Algorithm 1, searching radius R_s is a tradeoff between computation complexity and recovery quality. We let the size of block be 64, i.e., $n_b = 64$. Table 2 lists recovery time cost, recovery quality with different searching radius, and comparison with other sparse basis or recovery scheme. We can find from Table 2 that the larger R_s increases recovery quality, however, time cost is also increased. The experimental result shows that searching radius should be carefully chosen, especially when the frames of video have large size. Moreover, compared with fixed sparse basis, our sparse basis has the lowest time cost.

6.2. Evaluation on spatial domain layer encoding/decoding

At the spatial domain layer decoding, we first evaluate the cost of dictionary learning. In our experiment, sensor nodes capture videos with 4820 frames. We use first 800 frames to train sparse dictionary, and the rest frames for testing. Each frame has a resolution of 512×512, and block size is 8×8. Thus we get 4096 image blocks. To test the efficiency of dictionary learning, we use the sparsity of $S_i(k)$ as an indicator of performance. The higher sparsity of $S_i(k)$ means the better recovery quality or higher compression ratio. Fig. 9(a) shows that the number of non-zeros decreases, that is, the sparsity of vector $S_i(k)$ increases with the number of training samples. The red line shows that the best sparsity of vector $S_i(k)$ is 7 by our method, which can be reached around 180th frame, and most of blocks can reach the best sparsity before the first 550 frame. According to our experiment, each block reaches the best sparsity when there are 800 training frames. We regard 800 training frames as an acceptable cost, if we enable the spatial domain layer compression.

As stated in Section 5.2, we utilize the recovered image block $b_i^j(k)$ to assemble vector $B_i(k)$ and construct matrix $\mathscr{T}(k)$, then use $\mathscr{T}(k)$ to train sparse dictionary $D_*(k)$. Although this is more complex than training $D_*(k)$ directly by the encoded $Y_i(k)$, our indirect method provides better sparsity as shown in Fig. 9(a). If there are three sensor nodes in one cluster and the measurement ratio of temporal domain layer encoding is $\gamma_b = 0.25$, then the assembled vector $B_i(k)$ has 192 elements and the encoded vector $Y_*(k)$ has 48 elements. The blue line denotes the direct training method and shows that the sparsity of $S_i(k)$ is just reduced from 48 to 23. Obviously, our method provides better sparsity.

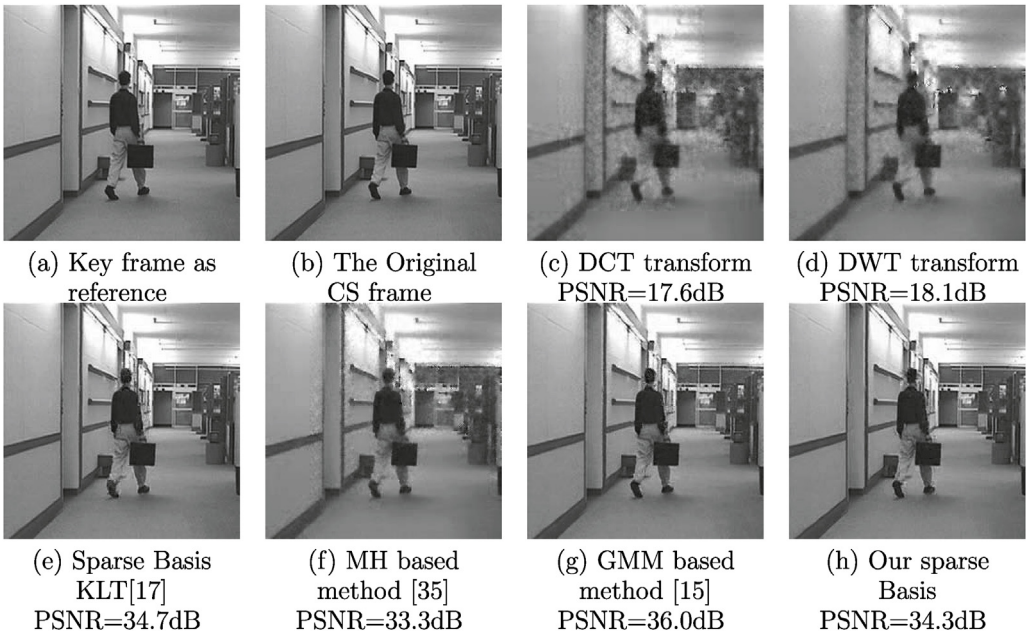


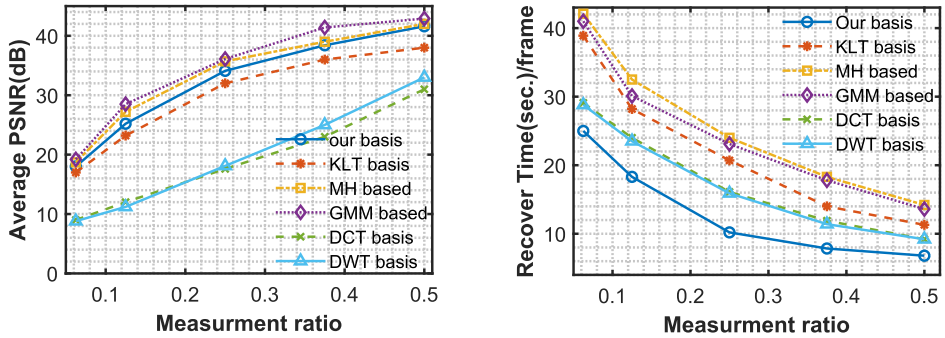
Fig. 7. Comparison of visual recovery quality with different sparse bases or recovery schemes. The key frame is measured at ratio $\Phi_k = 0.6$, and is recovered by Total-variation minimization. The CS frame is measured at ratio $\gamma_b = 0.25$. (c) and (d) are fix sparse basis. (e)–(h) exploit the correlation between frames, which provide better recovery quality.

In the following, we use the trained dictionary $D_*(k)$ to test the sparsity of $Y_*(k)$ which comes from $\mathscr{F}_{ts}$. Fig. 9(b) depicts the performance of sparsity. We choose 9 frames from $\mathscr{F}_{ts}$, and 4 assembled blocks from 4096 blocks in each frame. Fig. 9(b) shows that the assembled block *No*.1 has best sparse representation from the 801th frame to the 4801-th frame. Because the *No*.1 blocks of the three videos do not change in all frames, it can be sparsely represented by the trained dictionary. The same situation happens on the *No*.1500 block before the 2801-th frame. However, for some blocks, such as block *No*.3000 and *No*.4090, they cannot be well sparsely represented after the 1801th frame. The reason is that these blocks have intense variations among three videos. The result shows that, for some blocks, the sparse dictionary which is trained by early frames cannot well sparsely represent current frame. To solve this problem, we use adjacent frames of current frame to retrain sparse dictionary again. Thus in our frameworks, the dictionary learning of spatial domain layer is an online learning. When sparse representation of a specific block is getting worse, we extract recovered blocks in adjacent frames from different nodes, and assemble these blocks into a new $B_*(k)$. We then insert it into matrix $\mathscr{T}(k)$, and use the new $\mathscr{T}(k)$ to retrain dictionary $D_*(k)$ again. Because our framework is based on blocks, the cost of retraining process is very low.

6.3. Comparison on final reconstruction quality

In this subsection, we first compare the recovery quality of our two-layer framework with that of disparity/motion compensation schemes in (Liu et al., 2017; Trocan et al., 2014; Ebrahim and Chia, 2016). Our tests are based on two groups of videos. Fig. 10 shows the visual recovery quality of one group. In this group, two cameras capture the scene from different angles. The first row and second row show recovery quality of two cameras respectively. Fig. 10(a)(b) are final recovery result of our two-layer framework. Fig. 10(a) is based on $\gamma_b = 0.25$, $\gamma_s = 0.5$, and Fig. 10(b) is based on $\gamma_b = 0.3$ and $\gamma_s = 0.75$. Fig. 10(c)(d)(e) show the recovery results of multi-view by the methods in (Liu et al., 2017; Trocan et al., 2014; Ebrahim and Chia, 2016) respectively. The biggest difference between our method and others is that we employ CS twice, and the recovery of spatial domain is based on dictionary learning method. Fig. 10(b) shows that our framework provides similar recovery results with other three methods. Although our results are lower (2–5 dB) than other three schemes, our methods reduce 25% of total network traffic, due to that the spatial domain layer has measurement ration $\gamma_s = 0.75$. Fig. 10(a) has poor recovery quality because of $\gamma_s = 0.5$.

Fig. 11 shows the results of combination of different measurement ratios γ_b and γ_s for temporal domain layer and spatial domain layer. The



(a) Recovery quality of different decoding scheme or sparse basis.

(b) Recovery time for per CS frame of resolution:256 × 256.

Fig. 8. Comparison of recovery quality and time consumption for different bases or schemes.

Table 2
Comparison of time cost Frame/per and recovery quality.

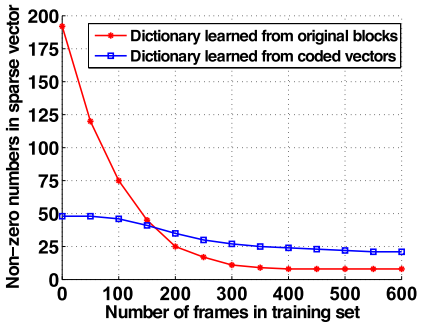
Performance		Our Sparse Basis $R_s = 2\sqrt{n_b}$	Our Sparse Basis $R_s = \sqrt{n_b}$	DCT	DWT	KLT	MH based	GMM based
PSNR (dB)	video1	40.7	38.1	21.6	21.2	39.5	38.3	40.6
	video2	38.2	36.7	18.7	19.1	37.7	36.9	37.9
	video3	39.5	37.7	17.8	17.6	38.1	37.7	39.1
	video4	40.9	39.0	23.1	22.1	39.5	39.2	40.7
Time (sec.)	video1	53.0	38.7	54.9	53.7	65.6	74.8	71.7
	video2	10.2	6.8	11.2	9.7	12.7	14.1	13.2
	video3	15.5	9.4	14.7	13.7	15.5	20.5	18.4
	video4	39.8	27.9	41.2	40.6	47.3	55.8	52.3

total measurement ratio is $\gamma_t = \gamma_b \times \gamma_s$, and the total compression ratio is $(1 - \gamma_t)$. $\gamma_s = 1$ in Fig. 11(a) denotes no encoding on the spatial domain layer, i.e., the spatial domain layer is deactivated. We observe from Fig. 11(a) that the final recovery quality reduces drastically as the measurement ratio γ_s in spatial domain layer decreases. Thus we must carefully choose the γ_s. For example, when we choose $\gamma_b = 0.25$ and $\gamma_s = 0.75$, the total measurement ratio $\gamma_t = 0.1875$ is less than that with $\gamma_b = 0.5$, $\gamma_s = 0.5$. However, we find the former has better recovery quality. The reason is that the recovery process in temporal domain layer encoding is based on the spatial domain layer recovery results and the slight errors in the spatial domain layer will lead to larger error of temporal domain layer encoding. Thus the final recovery quality is more sensitive to the spatial domain layer.

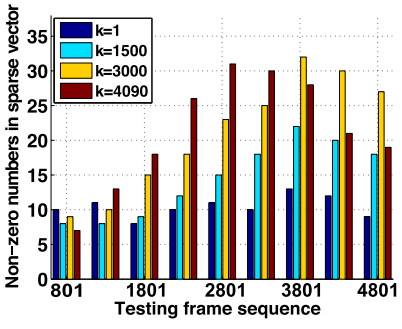
Fig. 11(b) shows the combination of two measurement ratios in 3D space and the principle of choosing good combination. We can find from Fig. 11(b) that the R-D function is a convex hull. In order for finding optimum combination of two measurement ratios, firstly, we need to decide the minimum recovery quality requirement, as shown in the cross section in Fig. 11(b). The upper combinations of the cross section are feasible. Furthermore, we need to choose the good combination with both high recovery quality and low measurement ratios. Note that since the two figures have different unit, we utilize the pair ($PSNR\#\gamma_t$) to label the good combination point in the 3D figure. For example, the point with label (30.1#0.1875) is a good combination, since it has the same compression ratio as the point(27.1#0.1875), and better recovery quality.

Next, we evaluate the sensitivity of our proposed sparse base with other reconstruction algorithms. We have compared 6 different reconstruction algorithms to solve $\|\cdot\|_1$ minimization problem. The six reconstruction algorithms (Yang, 2012) are listed as follows: GPSR (Gradient Projection Sparse recover Solver), OMP (Orthogonal Matching Pursuit Solver), L1LS (l1-Regularized Least Squares Problem Solver), PALM (Primal Augmented Lagrange Multiplier Solver), DALM (Dual Augmented Lagrange Multiplier Solver), BP (Basis Pursuit Solver). In this paper we employ BP reconstruction algorithm. Fig. 12 shows the visual recovery quality by different reconstruction algorithms. The total measurement ratio $\gamma_t = 0.2815$. The image size is 512×512. We can observe from Fig. 12 that the PSNRs of PALM, DALM, and BP algorithms are 34.9 db, 35.3 db, and 35.1 db, respectively, which are good recovery for such a low measurement ratio. Although the PSNRs of OMP and L1LS algorithms are slightly lower than others, they are still satisfying results. In summary, our sparse basis is robust to other $l1$ norm-based reconstruction algorithms.

Finally, we compare the rate distortion of our framework with the traditional compression method such as H.264 (Ahmed, 2016), and DVC (Zhuo et al., 2008) based method and another twice compression based method (Hirokawa et al., 2017). Recovery quality of DVC method is generally determined by encoding rate, while the recovery quality of our method is based on measurement ratio. We assume that sensor nodes employ 13-bit uniform scalar quantizer, measurement ratio is 0.25 and the resolution of video is 255×255 and FPS = 20. Although the data is always quantized with finite precision (8 bits), in fact the encoded data is not always quantized with fixed precision. For the encoded data, the precision of quantizer depends on maximum value of the encoded data. For example, we suppose an image block is the size of 8×8. After being vectorized, the image block becomes a vector b with



(a) Using original data to train dictionary is better than using encoded data.



(b) Using trained dictionary to test the sparsity of frames in $\mathcal{F}_{ts}$. The sparsity of the assembled blocks will decrease as frame sequence increases.

Fig. 9. The performance of trained dictionary.

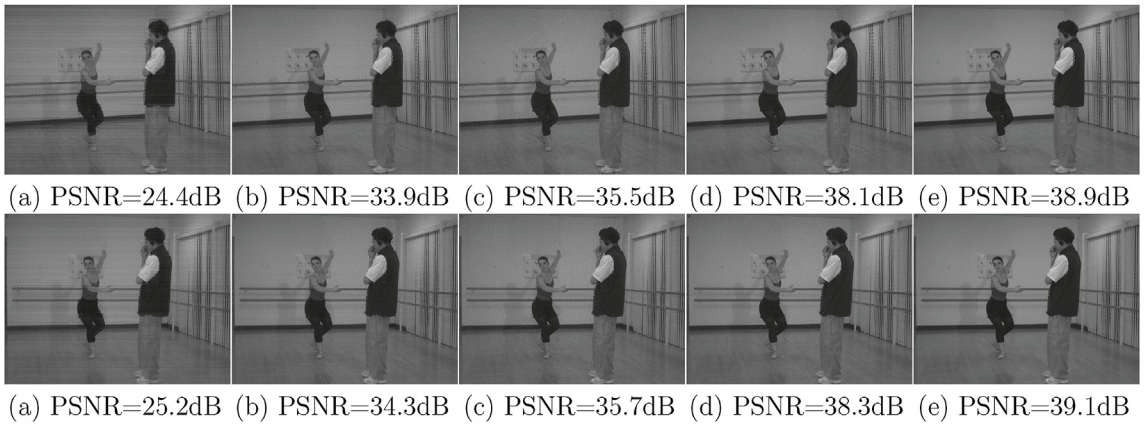
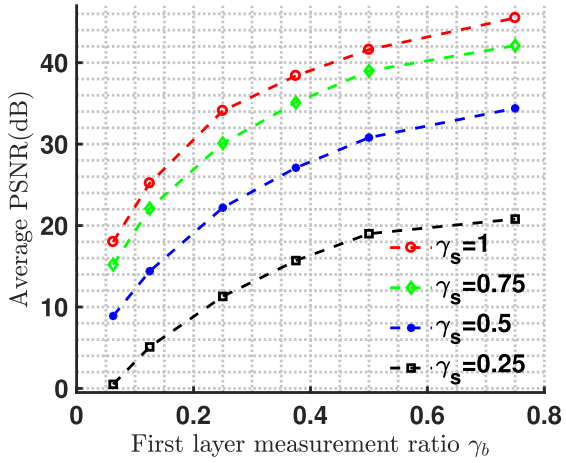
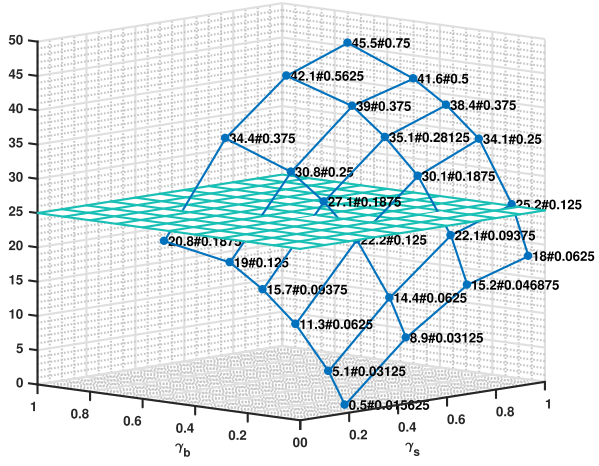


Fig. 10. Comparison of final reconstruction quality on multiple cameras. The first (a)–(e) figures are gotten by camera No.1, and the second (a)–(e) figures are gotten by camera No.2. They capture similar scene. (a)and (b) are final recovery result of our framework. (c) is CS based multiview video recovery with disparity and motion compensation (Liu et al., 2017). (d) is CS based multiview video recovery with signal prediction (Trocan et al., 2014). (e) Joint Multi-phase Decoding method (Ebrahim and Chia, 2016).



(a) Final recovery quality of two layers encoding. X-axis denotes the measurement ratio of temporal domain layer γ_b. Curves in the figure denote different measurement ratios of the spatial domain layer γ_s



(b) 3D demonstration about two layers measurement ratio in RD space. The recovered PSNR and total measurement ratio are labeled beside each point. The cross section denotes minimum recovery quality requirement.

Fig. 11. Comparison of combination of two layer measurement ratios.

size 64. Let ϕ denote random Gaussian matrix, where each item ϕ_{ij} ranges from 0 to 1. According to the definition of compressive sensing, we can get corresponding encoded vector $y_{16\times1} = \phi_{16\times64} \cdot b_{64\times1}$. It is clear that an item y_i in vector y equals to $y_i = \phi_{i^*} \cdot b$. An item in vector b represents a pixel. The maximum value of each item in vector b is 256. Thus the maximum value of all the items in vector y is $256 \times 64 = 2^{14}$. If we assume that the quantization is lossless, then the quantizer must be more than 14 bits. However, considering the ϕ_{ij} cannot always equal to 1, we find 13 bit-quantizer is enough for our reconstruction approach. This is why sensor nodes employ 13-bit uniform scalar quantizer in our experiment. In summary, the size of image block and the type of measurement matrix ϕ determine the bits of quantizer.

Fig. 13(a) shows the comparison results on recovery quality. It is not difficult to observe that our CS based framework has higher recovery quality than DVC and H.264 methods. Although the twice compression based method (Hirokawa et al., 2017) takes twice compression, our results are still superior to their results. The reason lies in the second compression in (Hirokawa et al., 2017) employs Wyner-Ziv encoding method. We notice that the H.264 method has high PSNR under high encoding rate. Thus, our CS based video coding/decoding framework has better PSNR under low encoding rate and is more suitable for WMSNs with low encoding rate scenario.

Fig. 13(b) shows that our method can save more energy than the traditional aggregation method (Sun et al., 2012), DVC method (Zhuo et al., 2008), H.264 method (Ahmed, 2016), twice compression method (Hirokawa et al., 2017). Generally, a camera samples 24 frames per second. After encoding and transmitting more than 250000 (about 3 h) frames, sensor nodes still have average 22.7% residual energy by our method. Compared with other methods, our twice encoding framework indeed prolongs the lifetime of whole network.

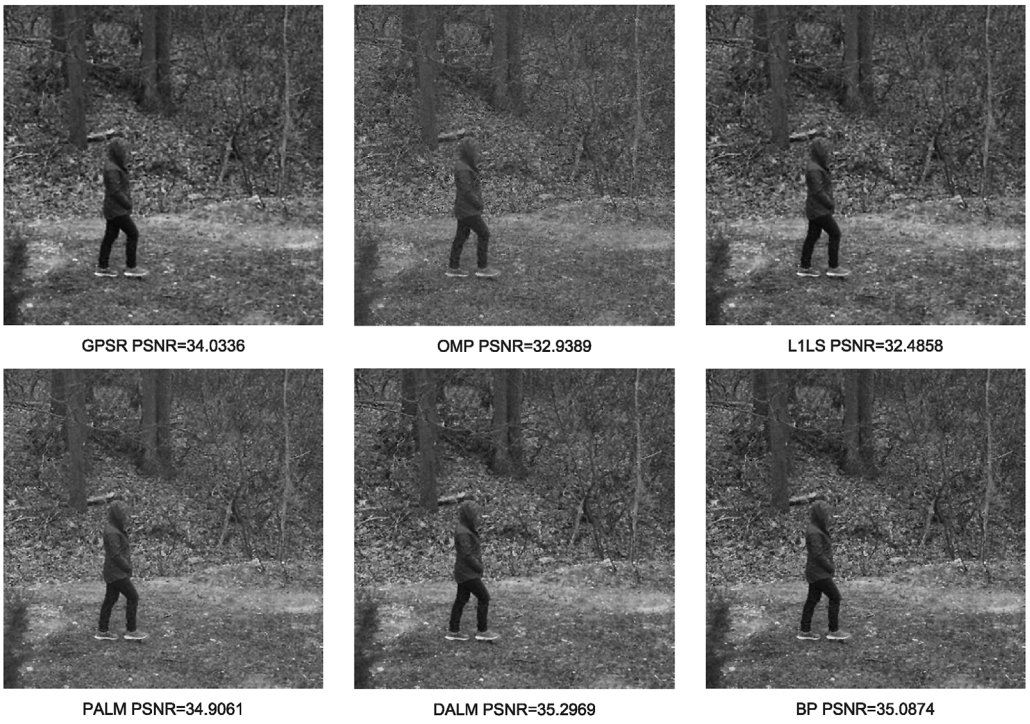
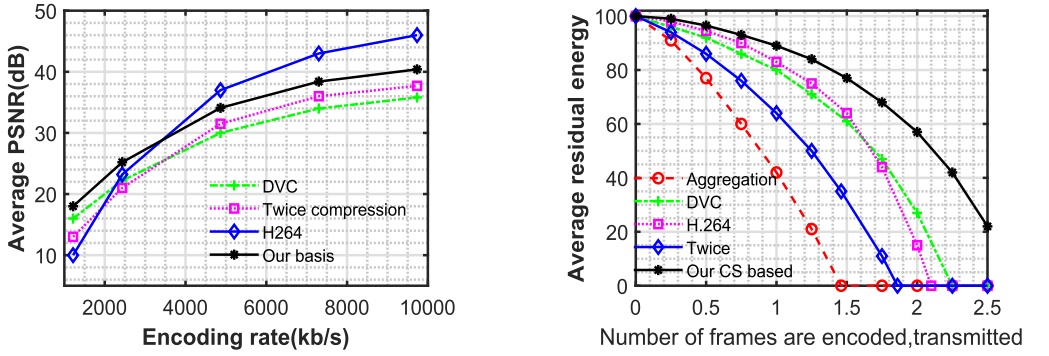


Fig. 12. The performance of our sparse bases with different reconstruction algorithms. The total measurement ratio $\gamma_t = 0.2815$.



(a) Rate distortion comparing with DVC and H.264.

(b) Residual energy after 250000 frames encoded and transmitted.

Fig. 13. Comparison of recovery quality and residual energy of sensor node.

7. Conclusions and future work

In this paper, we apply CS theory to encoding and compressing video in WMSNs. We propose a two-layer encoding/decoding framework. At each layer, sensor nodes or aggregation nodes just perform simple matrix multiplication. At sink node, to successfully decode spatial domain layer encoded data, we employ dictionary learning and key frame to construct two groups of sparse basis. Compared with existing state-of-the-art works, our sparse basis provides lower time cost while keeping good recovery quality. Our framework is flexible and provides higher compression ratio. Experimental results demonstrate that our framework reduces the amount of data packets in the transmission greatly, and prolongs the network lifetime significantly.

Since the compression ratio on the spatial domain layer is not high, in the future, we would like to use some filter methods to denoise the results of the spatial domain layer for improving final compression ratio and recovery quality. Data compression to reduce its volume is an effective method to reduce energy consumption for WMSNs. However, green communication and energy-harvesting (Yao et al., 2013) are other promising methods to prolong the lifetime of networks in recent years. In addition, multimedia applications consume more bandwidth than traditional WSNs. To gain more bandwidth and reduce delays, full-duplex technique and cognitive radio (Amjad et al., 2018) are also crucial approaches for WMSNs. We will integrate these paradigms into our further works.

Acknowledgement

This work was supported by National Natural Science Foundation of China (61772432, 61772433), Natural Science Key Foundation of Chongqing (cstc2015jcyjBX0094), Fundamental Research Funds for the Central Universities (XDJK2015D023, XDJK2016A011, and

XDJK2016D047), Natural Science Foundation of Chongqing (CSTC2016JCYJA0449), China Postdoctoral Science Foundation (2016M592619), and Chongqing Postdoctoral Science Foundation (XM2016002).

References

Ahmed, A.A., 2016. An optimal complexity h. 264/avc encoding for video streaming over next generation of wireless multimedia sensor networks. Signal, Image Video Process. 10 (6), 1143–1150.

Akyildiz, I.F., Melodia, T., Chowdury, K.R., 2007. Wireless multimedia sensor networks: a survey. Wireless Commun., IEEE 14 (6), 32–39.

Almalkawi, I.T., Guerrero Zapata, M., Al-Karaki, J.N., Morillo-Pozo, J., 2010. Wireless multimedia sensor networks: current trends and future directions. Sensors 10 (7), 6662–6717.

Amjad, M., Sharif, M., Afzal, M.K., Kim, S.W., 2016. Tinyos-new trends, comparative views, and supported sensing applications: a review. IEEE Sensor. J. 16 (9), 2865–2889.

Amjad, M., Rehmani, M.H., Mao, S., 2018. Wireless multimedia cognitive radio networks: a comprehensive survey. Spectrum 1, 2.

Asif, M.S., Fernandes, F., Romberg, J., 2013. Low-complexity video compression and compressive sensing. In: Signals, Systems and Computers, 2013 Asilomar Conference on. IEEE, pp. 579–583.

Azghani, M., Karimi, M., Marvasti, F., 2016. Multihypothesis compressed video sensing technique. IEEE Trans. Circ. Syst. Video Technol. 26 (4), 627–635.

Candes, E.J., Tao, T., 2005. Decoding by linear programming. IEEE Trans. Inf. Theor. 51 (12), 4203–4215.

Chen, C., Tramel, E.W., Fowler, J.E., 2011. Compressed-sensing recovery of images and video using multihypothesis predictions. In: Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on. IEEE, pp. 1193–1198.

Ciuonzo, D., Romano, G., Salvo Rossi, P., 2012. Channel-aware decision fusion in distributed mimo wireless sensor networks: decode-and-fuse vs. decode-then-fuse. IEEE Trans. Wireless Commun. 11 (8), 2976–2985.

Colonnese, S., Cuomo, F., Melodia, T., 2013. An empirical model of multiview video coding efficiency for wireless multimedia sensor networks. IEEE Trans. Multimed. 15 (8), 1800–1814.

Donoho, D.L., 2006. Compressed sensing. IEEE Trans. Inf. Theor. 52 (4), 1289–1306.

Ebrahim, M., Chia, W.C., 2016. Multiview image block compressive sensing with joint multiphase decoding for visual sensor network. ACM Trans. Multimed. Comput. Commun. Appl 12 (2), 30.

Ehsan, S., Hamdaoui, B., 2012. A survey on energy-efficient routing techniques with qos assurances for wireless multimedia sensor networks. IEEE Commun. Surv. Tutorials 14 (2), 265–278.

Elad, M., Aharon, M., 2006. Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans. Image Process. 15 (12), 3736–3745.

Eldar, Y.C., Kutyniok, G., 2012. Compressed Sensing: Theory and Applications. Cambridge University Press.

Hirokawa, S., Kurihara, S., Kikuchi, H., 2017. Distributed video coding based on compressive sensing and intra-predictive coding. In: Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017. IEEE, pp. 1701–1706.

Liu, Y., Pados, D.A., 2013. Decoding of framewise compressed-sensed video via interframe total variation minimization. J. Electron. Imag. 22 (2) 021012–021012.

Liu, Y., Li, M., Pados, D.A., 2013. Motion-aware decoding of compressed-sensed video. IEEE Trans. Circ. Syst. Video Technol. 23 (3), 438–444.

Liu, Y., Pados, D.A., Kim, J., Zhang, C., 2017. Reconstruction of compressed-sensed multiview video with disparity and motion compensated total-variation minimization. IEEE Trans. Circ. Syst. Video Technol. http://dx.doi.org/10.1109/TCSVT.2017.2656920. 1–1.

Medjiah, S., Ahmed, T., Asgari, A.H., 2012. Streaming multimedia over wmsns: an online multipath routing protocol. Int. J. Sens. Netw. 11 (1), 10–21.

Metzler, C.A., Maleki, A., Baraniuk, R.G., 2016. From denoising to compressed sensing. IEEE Trans. Inf. Theor. 62 (9), 5117–5144.

Mun, S., Fowler, J.E., 2011. Residual reconstruction for block-based compressed sensing of video. In: Data Compression Conference (DCC), 2011. IEEE, pp. 183–192.

Pan, Z., Zhang, Y., Kwong, S., 2015. Efficient motion and disparity estimation optimization for low complexity multiview video coding. IEEE Trans. Broadcast. 61 (2), 166–176.

Park, J.Y., Wakin, M.B., 2009. A multiscale framework for compressive sensing of video. In: Picture Coding Symposium, 2009. PCS 2009. IEEE, pp. 1–4.

Pudlewski, S., Prasanna, A., Melodia, T., 2012. Compressed-sensing-enabled video streaming for wireless multimedia sensor networks. IEEE Trans. Mobile Comput. 11 (6), 1060–1072.

Puri, R., Majumdar, A., Ishwar, P., Ramchandran, K., 2006. Distributed video coding in wireless sensor networks. IEEE Signal Process. Mag. 23 (4), 94–106.

Rashid, B., Rehmani, M.H., 2016. Applications of wireless sensor networks for urban areas: a survey. J. Netw. Comput. Appl. 60, 192–219.

Rein, S., Reisslein, M., 2011. Performance evaluation of the fractional wavelet filter: a low-memory image wavelet transform for multimedia sensor networks. Ad Hoc Netw. 9 (4), 482–496.

Salvo Rossi, P., Ciuonzo, D., Romano, G., 2013. Orthogonality and cooperation in collaborative spectrum sensing through mimo decision fusion. IEEE Trans. Wireless Commun. 12 (11), 5826–5836.

Salvo Rossi, P., Ciuonzo, D., Kansanen, K., Ekman, T., 2015. On energy detection for mimo decision fusion in wireless sensor networks over nlos fading. IEEE Commun. Lett. 19 (2), 303–306.

Sankaranarayanan, A.C., Studer, C., Baraniuk, R.G., 2012. Cs-muvi: video compressive sensing for spatial-multiplexing cameras. In: Computational Photography (ICCP), 2012 IEEE International Conference on. IEEE, pp. 1–10.

Shah, G.A., Liang, W., Akan, O.B., 2012. Cross-layer framework for qos support in wireless multimedia sensor networks. IEEE Trans. Multimed. 14 (5), 1442–1455.

Sun, Y., Luo, H., Das, S.K., 2012. A trust-based framework for fault-tolerant data aggregation in wireless multimedia sensor networks. IEEE Trans. Dependable Secure Comput. 9 (6), 785–797.

Thirumalai, V., Frossard, P., 2013. Joint reconstruction of multiview compressed images. IEEE Trans. Image Process. 22 (5), 1969–1981.

Tosic, I., Frossard, P., 2011. Dictionary learning. IEEE Signal Process. Mag. 28 (2), 27–38.

Trocan, M., Tramel, E.W., Fowler, J.E., Pesquet, B., 2014. Compressed-sensing recovery of multiview image and video sequences using signal prediction. Multimed. Tool. Appl. 72 (1), 95–121.

Usman, M., Yang, N., Jan, M.A., He, X., Xu, M., Lam, K.-M., 2018. A joint framework for qos and qoe for video transmission over wireless multimedia sensor networks. IEEE Trans. Mobile Comput. 17 (4), 746–759.

Wang, S., Wang, X., Ding, L., Bi, D., You, Z., 2008. Collaborative hybrid classifier learning with ant colony optimization in wireless multimedia sensor networks. In: Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on. IEEE, pp. 3341–3346.

Wu, M., Chen, C.W., 2007. Collaborative image coding and transmission over wireless sensor networks. EURASIP J. Appl. Signal Process. 2007 (1) 223–223.

Xie, X., Guan, L., Lu, Z., Lai, Z., 2009. Fast encoding of video based on compressive sensing. In: Information, Computing and Telecommunication, 2009. YC-ICT'09. IEEE Youth Conference on. IEEE, pp. 114–117.

Xiong, Z., Liveris, A.D., Cheng, S., 2004. Distributed source coding for sensor networks. IEEE Signal Process. Mag. 21 (5), 80–94.

Xu, H., Huang, L., Qiao, C., Zhang, Y., Sun, Q., 2012. Bandwidth-power aware cooperative multipath routing for wireless multimedia sensor networks. IEEE Trans. Wireless Commun. 11 (4), 1532–1543.

Yang, A.Y., 2012. L-1 Benchmark Package. https://people.eecs.berkeley.edu/~yang/software/l1benchmark/ [Online; accessed 19-July-2008].

Yang, J., Yuan, X., Liao, X., Llull, P., Brady, D.J., Sapiro, G., Carin, L., 2014. Video compressive sensing using Gaussian mixture models. IEEE Trans. Image Process. 23 (11), 4863–4878.

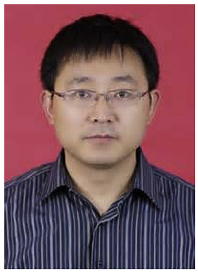
Yao, R., Wang, W., Farrokh-Baroughi, M., Wang, H., Qian, Y., 2013. Quality-driven energy-neutralized power and relay selection for smart grid wireless multimedia sensor based iots. IEEE Sensor. J. 13 (10), 3637–3644.

ZainEldin, H., Elhosseini, M.A., Ali, H.A., 2015. Image compression algorithms in wireless multimedia sensor networks: a survey. Ain Shams Eng. J. 6 (2), 481–490.

Zhuo, X., Loo, J., Cosmas, J., Yip, A., 2008. Distributed video coding in wireless multimedia sensor network for multimedia broadcasting. WSEAS Trans. Commun. 7 (5), 418–427.

Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R., 2004. High-quality video view interpolation using a layered representation. ACM Trans. Graph. 23 (3), 600–608.

Yang Yang received the BS and MS degree of computer science and engineering from the SouthWest University, China, in 2004 and 2008, respectively. He is currently working toward the PhD degree in the College of Electronic and Information Engineering, SouthWest University. His research interests include wireless sensor networks, resource allocation in cloud computation and data center.



Songtao Guo received the BS, MS, and PhD degrees in computer software and theory from Chongqing University, Chongqing, China, in 1999, 2003, and 2008, respectively. He was a professor from 2011 to 2012 at Chongqing University. He is currently a full professor at Southwest University, China. He was a senior research associate at the City University of Hong Kong from 2010 to 2011, and a visiting scholar at Stony Brook University, New York, from May 2011 to May 2012. His research interests include wireless networks, mobile cloud computing and parallel and distributed computing. He has published more than 80 scientific papers in leading refereed journals and conferences. He has received many research grants as a principal investigator from the National Science Foundation of China and Chongqing and the Postdoctoral Science Foundation of China.

Guiyan Liu received the B.S. degree in telecommunications engineering from Southwest University, Chongqing, China, in 2014. She is currently working toward the PhD's degree in signal and information processing, Southwest University. Her research interests include stream scheduling in data center networks and software defined networking.

Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a professor of computer engineering and computer science at Stony Brook University, New York. Her research interests include wireless networks, data center networks, optical networks and high-speed networks. She has published over 300 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She has served as an Associate Editor-in-Chief and an Associated Editor for IEEE Transactions on Computers and an Associate Editor for IEEE Transactions on Parallel and Distributed Systems. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is an IEEE Fellow.