Patch Tracking-Based Streaming Tensor Ring Completion for Visual Data Recovery

Yicong He and George K. Atia, *Senior Member, IEEE*

Abstract—Tensor completion aims to recover the missing entries of a partially observed tensor by exploiting its low-rank structure, and has been applied to visual data recovery. In applications where the data arrives sequentially such as streaming video completion, the missing entries of the tensor need to be dynamically recovered in a streaming fashion. Traditional streaming tensor completion algorithms treat the entire visual data as a tensor, which may not work satisfactorily when there is a big change in the tensor subspace along the temporal dimension, such as due to strong motion across the video frames. In this paper, we develop a novel patch tracking-based streaming tensor ring completion framework for visual data recovery. Given a newly incoming frame, small patches are tracked from the previous frame. Meanwhile, for each tracked patch, a patch tensor is constructed by stacking similar patches from the new frame. Patch tensors are then completed using a streaming tensor ring completion algorithm, and the incoming frame is recovered using the completed patch tensors. We propose a new patch tracking strategy that can accurately and efficiently track the patches with missing data. Further, a new streaming tensor ring completion algorithm is proposed which can efficiently and accurately update the latent core tensors and complete the missing entries of the patch tensors. Extensive experimental results demonstrate the superior performance of the proposed algorithms compared with both batch and streaming state-of-the-art tensor completion methods.

Index Terms—Tensor completion, tensor ring decomposition, streaming video processing.

I. INTRODUCTION

MULTI-WAY data analysis uses techniques that represent data as multi-dimensional arrays known as tensors. It has drawn increased attention in recent years given its ability to reveal patterns intrinsic to high-order data undetected by other methods by capturing correlations across its different dimensions. Such techniques have found numerous applications in machine learning [1]–[5], signal processing [6]–[8] and computer vision [9]–[13].

Tensor completion, an extension to the matrix completion problem, aims to fill in the missing entries of a partially observed tensor by leveraging its low-rank structure, which stems from the redundancy of the information it represents [14], [15]. For example, a natural multi-channel image or a video sequence can be represented as tensors that exhibit high correlations across the spatial and temporal dimensions. Based on different tensor rank models, several completion algorithms have been proposed, such as CP-based algorithms [16], [17], tucker-based algorithms [18]–[20], tubal (t-SVD)-based algorithms [21]–[23], tensor train-based algorithms [10], [24] and tensor ring-based algorithms [25]–[27]. Extensive experimental results of these tensor completion algorithms have shown their superior completion performance compared to matrix completion.

Traditional batch tensor completion methods are of limited use in streaming applications wherein the data arrives sequentially, since they are designed to work on the entire tensor data. To mitigate this problem, several methods have been proposed to complete the missing entries of data in a streaming or online manner. In [28], the authors developed an online tensor completion approach using the CP model and stochastic gradient descent (SGD). The authors in [29] proposed an online low-rank subspace tracking by tensor CP decomposition (OLSTEC), which improves the efficiency and accuracy of [28] by utilizing recursive least squares (RLS) instead of SGD. Then, [30] developed a CP-based multi-aspect streaming tensor completion considering the case that the tensor grows along multiple modes. Further, a tensor rank-one update on the complex Grassmannian (TOUCAN) algorithm is proposed under the t-SVD model [31]. Also, a sequential tensor completion (STC) approach was developed for the recovery of internet traffic data under the Tucker tensor model [32].

Despite the success of these streaming methods in completing the missing entries of sequential data, there are important limitations that remain to be addressed. In particular, all existing methods require that the entire tensor subspace should be stable, that is, the latent tensor factors should be invariant or changing slowly along the temporal dimension. In the presence of large motion between adjacent tensors, the subspace may change considerably, resulting in degraded completion performance. As shown in the first row of Fig.1, some objects (car, road sign) move significantly across different frames. Such large motions increase the rank of the mode-4 unfolding matrix (see bottom right of Fig. 1), indicating a big change in the tensor subspace. Thus, existing streaming tensor completion algorithms may not accurately complete the objects with large motion.
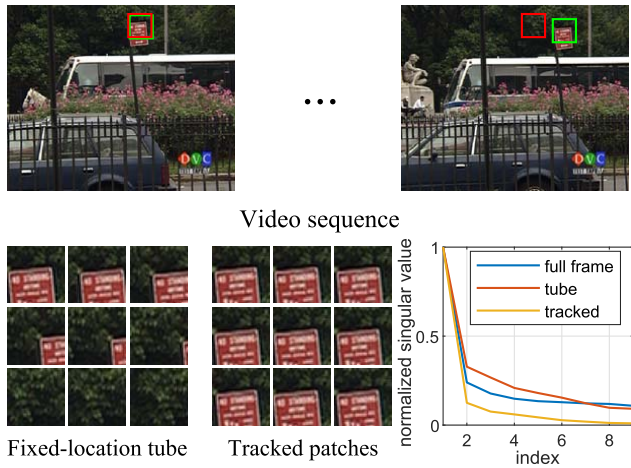
Fig. 1. Illustration of the patch tracking-based method for the video sequence in the top row. Starting from the first frame, the green patches are tracked along the temporal dimension while the positions of red patches are kept unchanged. The bottom row shows the extracted patches, and the normalized singular values (i.e., each singular value divided by the largest singular value) of the mode-4 unfolding matrix of the tensor formed by stacking the full frames, fixed-location patches (tube), and tracked patches along the fourth (temporal) dimension.

In this paper, we introduce a novel patch tracking-based streaming tensor ring completion framework for visual data recovery (see Fig. 2). Rather than performing completion on the entire incoming frame, we complete small patch tensors constructed from similar patches extracted and tracked from the frame. As a new frame comes in, similar patches across the temporal domain are tracked using patch matching. For example, as shown in Fig. 1, the patches of 'road sign' are tracked. Compared with fixed tube and full-frame tensors, the normalized singular values of the mode-4 unfolding matrix of the tracked patches are relatively small, indicating a more stable tensor subspace along the temporal domain. Therefore, the completion performance can be improved substantially when applied to similar tracked patches.

Patch-based methods have been widely applied to visual data processing, such as image and video denoising [33]–[36]. However, patch matching becomes more challenging when the data has missing entries, due to insufficient matching information. The authors of [37] address the problem of patch matching with missing entries by interpolating images whose missing pixels are filled using triangle interpolation. An alternative approach is used in [38] where the missing pixels are estimated by applying tensor completion before matching. These interpolation or completion methods, however, incur more computational cost and may introduce additional errors to patch matching. The authors in [39] proposed a tube matching approach by adding the information of the temporal domain as an extra dimension. The tubes are matched along the spatial domain and completion is performed on the stacked tubes. However, tube matching is a batch method and is not applicable to streaming tensor completion since the information in the temporal domain is not available. Further, as Fig.1 depicts, tube matching ignores the change in motion along the temporal dimension. In this work, we develop a

novel efficient strategy for patch matching. We perform patch matching on dilated tensor data by propagating the observed pixels to unobserved neighboring pixels. Then, a new efficient coarse-scale patch matching strategy is proposed to efficiently match the patches with missing data.

To accurately complete the missing entries of a streaming tensor formed from tracked patches, we develop a new streaming tensor completion algorithm using the tensor ring model. This model exploits a more compact expression for high-order tensors, and has shown superior performance in tensor completion, especially with highly incomplete data [40]. Compared to CP-based streaming tensor completion methods [28], which are only designed for 3-way tensors, the proposed streaming tensor ring completion algorithm can work with higher-order tensors required by the proposed framework. Compared with TOUCAN [31] and STC [32], the proposed method utilizes the more compact latent core tensors to track the tensor subspace. We propose a fast streaming completion algorithm for the tensor ring model using a least-squares and one-step scaled steepest descent method, which can efficiently update the latent core tensors and complete the streaming tensor with high accuracy.

The main contributions of the paper are summarized as follows:

1. We develop a novel patch tracking-based streaming tensor ring completion framework for visual data recovery. An efficient coarse-scale patch matching strategy is proposed for patch tracking with missing data. To the best of our knowledge, this is the first patch-based framework that incorporates a streaming tensor completion method.

2. We propose a new streaming tensor completion algorithm using the tensor ring model. The algorithm utilizes least-squares and one-step scaled steepest descent, and can efficiently update the latent core tensors and complete the tracked patch tensor.

3. We conduct extensive experiments that demonstrate the superior performance of the proposed method over state-of-the-art batch and streaming tensor completion algorithms.

The paper is organized as follows. In Section II, we briefly introduce our notation and provide some preliminary background on the tensor ring and its properties. In Section III, we propose the new patch-based streaming tensor ring completion framework. In Section IV, we develop the streaming tensor ring completion algorithm. The complexity analysis is given in Section V. Experimental results are presented in Section VI to demonstrate the completion performance. Finally, the conclusion is given in Section VII.

II. PRELIMINARIES

Uppercase script letters are used to denote tensors (e.g., $\mathcal{X}$), boldface uppercase letters to denote matrices (e.g., $\mathbf{X}$) and boldface lowercase letters to denote vectors (e.g., $\mathbf{x}$). An N-order tensor is defined as $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, where $I_i, i \in [N] := \{1, \dots, N\}$ is the dimension of the i-th way of the tensor, and $\mathcal{X}_{i_1 \dots i_N}$ denotes the $(i_1, i_2, \dots, i_N)$-th entry of tensor $\mathcal{X}$. The vector $\mathbf{x} \in \mathbb{R}^{\prod_{k=1}^N I_k}$ denotes the vectorization of tensor $\mathcal{X}$. For a 3-way order tensor (i.e., $N = 3$),
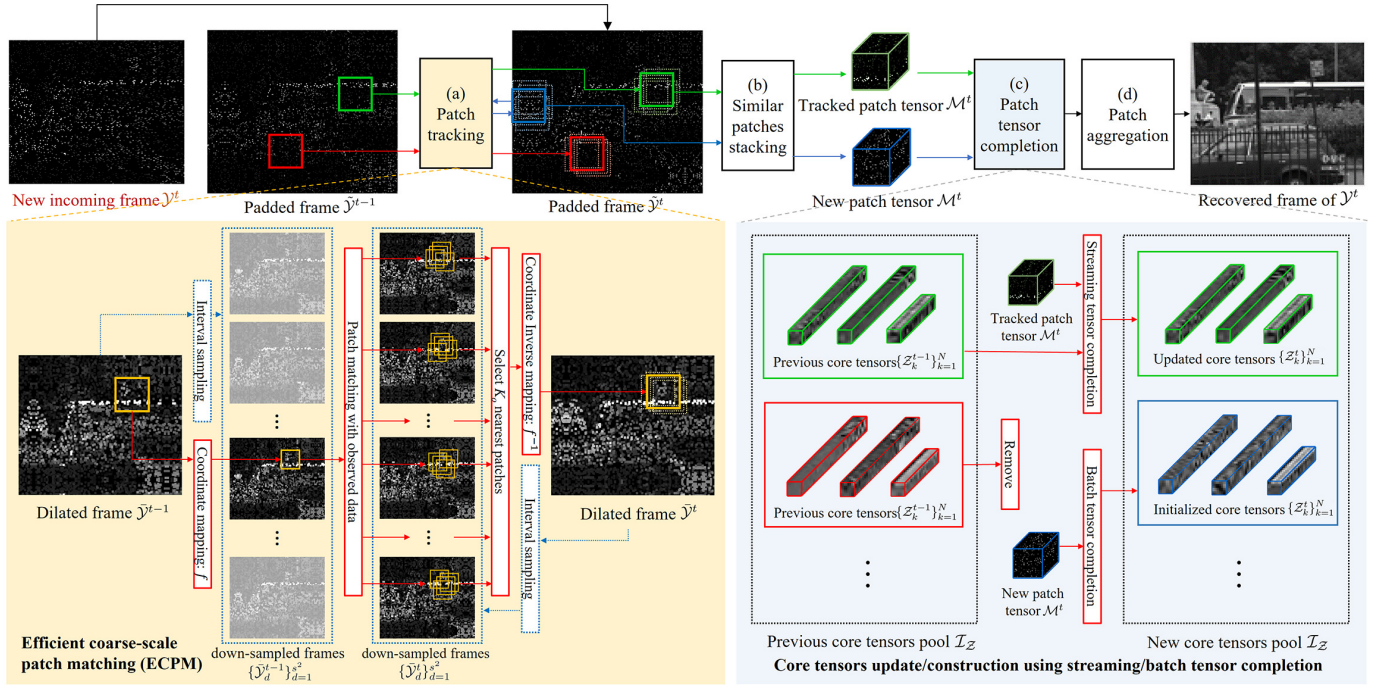
Fig. 2.   Illustration of the proposed patch tracking-based streaming tensor ring completion framework. Given a newly incoming frame $\mathcal{Y}^t$, (a) an efficient patch tracking method is firstly applied to track the patches from frame $\mathcal{Y}^{t-1}$. Specifically, for tracked patches of $\mathcal{Y}^{t-1}$ (shown as green and red solid rectangles), K_o most similar patches in $\mathcal{Y}^t$ are matched using the proposed ECPM method (the first similar patch with smallest distance is shown as a solid rectangle, while the remaining $K_o - 1$ patches are shown as dotted rectangles). The blue solid rectangle designates a new patch for an uncovered region in $\mathcal{Y}^t$, and the dotted blue rectangles around it denote its $K_b - 1$ most similar patches. (b) Patch tensors are created by stacking K_o (or K_b) nearest similar green (or blue) patches. (c) Core tensors pool is updated using the corresponding patch tensors, and subsequently patch tensors are completed using the updated core tensors pool. Here, we assume that the red tracked patch in $\mathcal{Y}^t$ is mistracked or has large overlap with other tracked patches, and should be removed from the tracking process. (d) Finally, the recovered new frame is obtained by aggregating the completed patch tensors.

the notation $\mathcal{X}(:, :, i)$, $\mathcal{X}(:, i, :)$, $\mathcal{X}(i, :, :)$ denotes the frontal, lateral, horizontal slices of $\mathcal{X}$, respectively. The Frobenius norm of a tensor $\mathcal{X}$ is defined as $\|\mathcal{X}\|_F = \sqrt{\sum_{i_1 \ldots i_N} |\mathcal{X}_{i_1 \ldots i_N}|^2}$. The product $\mathcal{A} \circ \mathcal{B}$ denotes the Hadamard (element-wise) product of two tensors $\mathcal{A}$ and $\mathcal{B}$. Next, we introduce some definitions and theorems for tensor ring used throughout the paper.

Definition 1 (TR Decomposition [41]): In TR decomposition, a high-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ can be represented using a sequence of circularly contracted 3-order core tensors $\mathcal{Z}_k \in \mathbb{R}^{r_k \times I_k \times r_{k+1}}, k = 1, \ldots, N, r_{N+1} = r_1$. Specifically, the element-wise relation of tensor $\mathcal{X}$ and its TR core tensors $\{\mathcal{Z}_k\}_{k=1}^N$ is defined as

$$\mathcal{X}_{i_1 \ldots i_N} = \mathrm{Tr}\left(\prod_{k=1}^N \mathcal{Z}_k(:, i_k, :)\right), \qquad (1)$$

where $\mathrm{Tr}(\cdot)$ is the matrix trace operator. The relation of tensor $\mathcal{X}$ and its TR decomposition core tensors $\{\mathcal{Z}_k\}_{k=1}^N$ is written as $\mathcal{X} = \Re(\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_N)$, where $\Re$ is the function defined through (1), and $[r_1, \ldots, r_N]$ is called TR rank.

Definition 2 (Mode-k Unfolding [40], [42]): The mode-k unfolding matrix of tensor $\mathcal{X}$ is denoted by $\mathbf{X}_{[k]}$ of size $n_k \times \prod_{j \neq k} n_j$ with its elements defined by

$$X_{[k]}\left(i_k, \overline{i_{k+1}, \ldots, i_N i_1, \ldots, i_{k-1}}\right) = \mathcal{X}(i_1, i_2, \ldots, i_N) .$$

Another classical mode-k unfolding matrix of $\mathcal{X}$ is denoted by $\mathbf{X}_{(k)}$ of size $n_k \times \prod_{j \neq k} n_j$ and defined by

$$X_{(k)}\left(i_k, \overline{i_1, \ldots, i_{k-1} i_{k+1}, \ldots, i_N}\right) = \mathcal{X}(i_1, i_2, \ldots, i_N) .$$

Theorem 1 [41]: Given a TR decomposition $\mathcal{X} = \Re(\mathcal{Z}_1, \ldots, \mathcal{Z}_N)$, its mode-k unfolding matrix $\mathbf{X}_{[k]}$ can be written as

$$\mathbf{X}_{[k]} = \mathbf{Z}_{k(2)} \left(\mathbf{Z}_{[2]}^{\neq k}\right)^T$$

where $\mathbf{Z}^{\neq k} \in \mathbb{R}^{r_{k+1} \times \prod_{j=1, j \neq k}^N I_j \times r_k}$ is a subchain obtained by merging all cores except $\mathcal{Z}^k$, whose slice matrices are defined by

$$\mathbf{Z}^{\neq k}\left(\overline{i_{k+1} \cdots i_N i_1 \ldots i_{k-1}}\right) = \prod_{j=k+1}^N \mathcal{Z}_j(:, i_j, :) \prod_{j=1}^{k-1} \mathcal{Z}_j(:, i_j, :) .$$

III. PATCH TRACKING-BASED STREAMING TENSOR RING COMPLETION FRAMEWORK

In this section, we propose our patch tracking-based streaming tensor ring completion (PTSTRC) framework for streaming visual data recovery, which is illustrated in Fig. 2. First, we start with a high-level description of the proposed framework, followed by a detailed description of the different procedures used.

Assume we have a partially observed video sequence $\{\mathcal{Y}^i\}_{i=1}^T$, $\mathcal{Y}^i \in \mathbb{R}^{I_1 \times I_2 \times n}$, with corresponding observed pixel index sets $\{\Omega^i\}_{i=1}^T$. The number of frames T can be sufficiently large such that the video is a streaming video. For gray-scale and color frames, the value of n is 1 and 3, respectively. The video frames are given sequentially, i.e., the video frame $\mathcal{Y}^t$ arrives at time t. To match the patches at the corners and boundaries of the frames, the arriving frame $\mathcal{Y}^t$ is first padded by mirroring b pixels at all boundaries and corners, resulting in a padded frame $\tilde{\mathcal{Y}}^t$ with frame size $(I_1 + 2b) \times (I_2 + 2b) \times n$ and corresponding observation index sets $\tilde{\Omega}^t$.

For the initial video frame $\mathcal{Y}^1$, after padding it to $\tilde{\mathcal{Y}}^1$, we divide it into overlapping patches of size $m \times m \times n$ with a number of overlap pixels o, and set each of these patches as a 'tracked patch' of $\tilde{\mathcal{Y}}^1$. For each tracked patch, we find the $K_b - 1$ most similar patches within a search window of size $l \times l$ centered at the location of the tracked patch. Then, a patch tensor of size $m \times m \times n \times K_b$ is created by stacking the tracked patch and its $K_b - 1$ similar patches.

At time $t > 1$, for each tracked patch of $\tilde{\mathcal{Y}}^{t-1}$, a patch matching method is used to find the K_o most similar patches in $\tilde{\mathcal{Y}}^t$ within a search window of size $l \times l$. Meanwhile, the first similar patch (i.e., the one with the smallest distance) is designated as a tracked patch of $\tilde{\mathcal{Y}}^t$ and will be utilized for patch tracking of the next incoming frame. Then, a tracked patch tensor of size $m \times m \times n \times K_o$ is formed by stacking the K_o most similar patches. Further, patch refinement is invoked to prune the mistracked or highly overlapping tracked patches, as well as create new patches for uncovered regions.

After obtaining all the patch tensors required for completion of $\tilde{\mathcal{Y}}^t$, the relevant core tensors for each patch tensor are updated (or created) using a streaming (or batch) tensor ring completion algorithm. Then, the core tensors are stored in a tensor pool for completion of the next frame. Finally, the frame at time t is recovered as $\mathcal{P}^t \circ \mathcal{Y}^t + (1 - \mathcal{P}^t) \circ \hat{\mathcal{Y}}^t$, where the observation mask $\mathcal{P}^t$ is defined by

$$\mathcal{P}^t_{i_1,i_2,i_3} = \begin{cases} 1, & \text{if } (i_1, i_2, i_3) \in \Omega^t \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

and $\hat{\mathcal{Y}}^t$ is obtained by aggregating all the recovered patches belonging to $\tilde{\mathcal{Y}}^t$ and removing padded border pixels.

As shown in Fig. 2, the framework contains two main procedures, namely, patch tracking and patch tensor completion. Next, we discuss the different components underlying our patch tracking procedure. streaming tensor completion will be discussed in the next section.

A. Image Dilation for Patch Tracking With Missing Data

Compared with traditional patch tracking methods [33]–[36] where the data is fully observed, patch tracking with a portion of observed data is more challenging, especially when the number of observed pixels is relatively small. For example, for a patch of size 20×20 and observation ratio 20%, the number of observed pixels common between two patches will be about 16 in average. In this case, the matching accuracy may degrade due to lack of enough common information. In this paper, we address this problem using a simple but efficient image

dilation method. Specifically, the (i, j)-th entry of the dilated frame $\bar{\mathcal{Y}}$ for the n-th channel is given by

$$\bar{\mathcal{Y}}_{i,j,n} = \begin{cases} \max_{(p,q,n) \in \mathcal{N}_{i,j,n}, (p,q,n) \in \tilde{\Omega}} \tilde{\mathcal{Y}}_{p,q,n}, & \mathcal{N}_{i,j,n} \cap \tilde{\Omega} \neq \varnothing \\ \text{NaN}, & \text{otherwise} \end{cases} \quad (3)$$

where $\mathcal{N}_{i,j,n}$ is the index set formed by 9 neighbors of index (i, j) in the n-th channel (including itself), and NaN denotes that the corresponding entry is missing.

There are two advantages in using dilation. First, the observed pixels are propagated to the unobserved neighboring pixels, so the observation percentage is greatly improved. Second, the maximization operation among adjacent observed pixels brings larger positional margin to matching, which will be useful for the next coarse-scale matching step. We remark that using a dilated image for matching may yield patches that are within a small offset from the true matching positions. However, since it is a small offset, it does not affect the completion performance significantly, which will be verified in the experiments.

B. Efficient Coarse-Scale Patch Matching

Another bottleneck for the patch-based method is the computational efficiency. Searching similar patches could take a long time if the size of the search window is large. On the other hand, a small search window size may decrease the matching performance. To improve the efficiency of patch matching, one could make use of a hierarchical or multi-resolution search method [43], [44], which searches for similar patches on different coarse to fine scales. However, due to the existence of missing entries, the traditional interpolation-based downscale method [45] cannot be applied to the partially observed frames, which precludes direct use of the hierarchical search method for our matching task.

To improve the matching efficiency, in this work we propose a new efficient coarse-scale patch matching (ECPM) method, illustrated in the bottom-left of Fig. 2. At time t, given a sampling interval parameter s, both dilated new frame $\bar{\mathcal{Y}}^t$ and previous frame $\bar{\mathcal{Y}}^{t-1}$ are down-sampled by selecting pixels with interval s. Then, the corresponding s^2 down-sampled frames $\{\bar{\mathcal{Y}}^t_d\}_{d=1}^{s^2}$ and $\{\bar{\mathcal{Y}}^{t-1}_d\}_{d=1}^{s^2}$ of size $(I_1+2b)/s \times (I_2+2b)/s$ are obtained (here we assume that b is such that $I_1 + 2b$ and $I_2 + 2b$ are divisible by s). Patch matching is performed on the down-sampled frames and finally the locations of the matched patches in $\bar{\mathcal{Y}}^t$ are obtained. The detailed steps of the proposed ECPM method are described next. To simplify the expressions, we use the coordinate of the top left pixel of the patch to represent the patch location.

1) Build Pixel-Wise Coordinate Mapping Between Original Frame and Down-Sampled Frames:
The mapping function $f : \mathbb{R}^2 \to \mathbb{R}^3$ is defined as $f : (x, y) \mapsto (x', y', c)$ where

$$x' = \left\lfloor \frac{x - 1}{s} \right\rfloor + 1, \quad y' = \left\lfloor \frac{y - 1}{s} \right\rfloor + 1,$$
$$c = \text{mod}(x - 1, s) + \text{mod}(y - 1, s) \times s + 1 \quad (4)$$

with $c \in [1, s^2]$, such that the pixel location (x, y) in the original image is mapped to location (x', y') in the c-th down-sampled frame.

2) Perform Patch Matching on Down-Sampled Frames: Given the tracked patch located at (x_0, y_0) in the previous frame $\bar{\mathcal{Y}}^{t-1}$, we obtain its location (x_0', y_0') in the c_0-th down-sampled frame $\bar{\mathcal{Y}}_{c_0}^{t-1}$ using (4). Then, the patch with size $\lceil m/s \rceil \times \lceil m/s \rceil \times n$ at (x_0', y_0') in $\bar{\mathcal{Y}}_{c_0}^{t-1}$ is utilized as the down-sampled tracked patch, and patch matching is performed across all the s^2 down-sampled frames $\{\bar{\mathcal{Y}}_d^t\}_{d=1}^{s^2}$ of $\bar{\mathcal{Y}}^t$ within a search region of size $\lceil l/s \rceil \times \lceil l/s \rceil$ centered at (x_0', y_0'). In particular, given two patches $\mathcal{S}_1$ and $\mathcal{S}_2$ with corresponding observation masks $\mathcal{P}_1$ and $\mathcal{P}_2$, the distance (dissimilarity) is measured as

$$d(\mathcal{S}_1, \mathcal{S}_2) = \frac{\|\mathcal{P}_1 \circ \mathcal{P}_2 \circ (\mathcal{S}_1 - \mathcal{S}_2)\|_F^2}{\|\mathcal{P}_1 \circ \mathcal{P}_2\|_0} \qquad (5)$$

where the ℓ_0-norm $\|\cdot\|_0$ counts the total number of non-zero elements. A larger distance d indicates smaller similarity. The K_o most similar patches of the tracked patch in $\{\bar{\mathcal{Y}}_d^t\}_{d=1}^{s^2}$ are obtained with locations $\{(x_i', y_i', c_i)\}_{i=1}^{K_o}$.

3) Map the Matched Patches to Original Location: The locations of the K_o most similar patches in the new frame $\bar{\mathcal{Y}}^t$ (and $\tilde{\mathcal{Y}}^t$) can be computed using the following inverse mapping $f^{-1} : (x', y', c) \mapsto (x, y)$ where

$$\begin{aligned} x &= (x' - 1)s + 1 + \mathrm{mod}(c - 1, s) \\ y &= (y' - 1)s + 1 + \left\lfloor \frac{c - 1}{s} \right\rfloor, \end{aligned} \qquad (6)$$

Note that patch matching on each down-sampled frame is independent, so parallelization could be used to further improve efficiency. Using the above ECPM method, for each tracked patch in $\tilde{\mathcal{Y}}^{t-1}$, the K_o most similar patches in the new frame $\tilde{\mathcal{Y}}^t$ are obtained and stacked to construct a patch tensor of size $m \times m \times n \times K_o$. Specifically, patch matching can be applied within the frame $\tilde{\mathcal{Y}}^{t-1}$ itself by setting the new frame as $\tilde{\mathcal{Y}}^{t-1}$. We remark that the coarse-scale matching may also introduce a small offset between the tracked position and the true matching positions. However, as verified in the experiments, such a method achieves significant speedup with only slight degradation in recovery performance.

C. Patch Refinement

During the patch tracking process, there are three types of patches to be processed at a given time t, marked in green, red and blue in Fig. 2. In particular, correctly tracked patches (solid green rectangle in $\tilde{\mathcal{Y}}^t$) will be used for completion of $\tilde{\mathcal{Y}}^t$ and tracking of the next frame $\tilde{\mathcal{Y}}^{t+1}$. Some mistracked or highly overlapping patches (solid red rectangle in $\tilde{\mathcal{Y}}^t$) should be removed from the tracking procedure. Further, new patches (solid blue rectangle in $\tilde{\mathcal{Y}}^t$) should be created to cover regions that are not covered by the tracked patches. The specific operations for the latter two types of patches are detailed next.

1) Pruning Mistracked or Highly Overlapping Patches: Long video data may contain scene variations or abrupt changes between adjacent frames. In this case, patch tracking may result in tracking incorrect patches with low similarity. Therefore, such mistracked patches should be removed from the tracking procedure. To this end, we set a threshold τ_f

to decide if the patch tracking is 'success' or 'failure'. For a tracked patch of $\tilde{\mathcal{Y}}^t$, if its distance to the corresponding tracked patch in $\tilde{\mathcal{Y}}^{t-1}$ is larger than τ_f, this patch tracking process is deemed a failure and the tracked patch is removed. The corresponding core tensors in the core tensors pool are also removed.

It can be also observed that tracked patches may have considerable overlap after several tracking iterations. Continuing to track all such patches will result in no or marginal improvement in the completion performance at the expense of extra computational and storage cost. Thus, we stop tracking of these patches by detecting their degree of overlap. Specifically, the overlap degree of a tracked patch is obtained by firstly counting the number of times each pixel is shared with other tracked patches, and then picking the minimum number. If the overlap degree is larger than a predefined threshold τ_c, the corresponding tracked patch is removed.

2) Generating New Patches for Uncovered Regions: Some regions in the new frame $\tilde{\mathcal{Y}}^t$ may not be covered by tracked patches from $\tilde{\mathcal{Y}}^{t-1}$ due to the emergence of new scenes or the removal of mistracked patches. We create new overlapping patches of size $m \times m \times n$ with a number of overlap pixels o to cover these regions. These new patches are added to the tracked patches of $\tilde{\mathcal{Y}}^t$. Then, for each new patch, the ECPM method is applied to find the $K_b - 1$ most similar patches in $\tilde{\mathcal{Y}}^t$ around it within a search window of size $l \times l$, and a new patch tensor of size $m \times m \times n \times K_b$ is created by stacking the new patch and its $K_b - 1$ similar patches. Finally, a batch tensor completion algorithm is applied to the new patch tensors and the obtained core tensors are added to the core tensors pool.

The entire process of the proposed PTSTRC method at time t is summarized in Algorithm 1. It should also be noted that, for tracked patch set $\mathcal{I}_\mathcal{S}$, only the patch location information need to be stored. Further, one can recover $\mathcal{Y}^1$ at $t = 1$ by setting $\mathcal{I}_\mathcal{S} = \mathcal{I}_\mathcal{Z} = \varnothing$, i.e., treating $\mathcal{Y}^1$ as a frame with an entirely uncovered region.

According to Algorithm 1, to complete the new frame $\mathcal{Y}^t$, in practice one should store the previous frame $\mathcal{Y}^{t-1}$ with the corresponding observed pixel index set Ω^{t-1}, the coordinates of tracked patches of $\tilde{\mathcal{Y}}^{t-1}$, and the core tensors corresponding to the tracked patches. Taking advantage of the TR decomposition model, the past information is fused into the core tensors so the algorithm can run in a streaming fashion without storage of frames before $t - 1$.

Remark 1: Algorithm 1 can be simplified to a patch matching (PM)-based image completion algorithm, where only one incoming frame (image) is given and needs to be completed. In this case, step 2 to step 9 are removed (no tracking needed). For step 13, the completion is no longer limited to tensor-decomposition-based algorithms since there is no need to construct or update core tensors. For example, $\mathcal{M}_i^t$ can be completed using nuclear-norm-based algorithms such as tensor ring nuclear norm (TRNN) [42] and tensor train nuclear norm (TTNN) [10]. However, nuclear-norm-based algorithms are not applicable to streaming tensor completion with sequential incoming frames because they cannot construct and update core tensors.

Algorithm 1 Patch Tracking-Based streaming Tensor Ring Completion (PTSTRC)

Input: New incoming frame $\mathcal{Y}^t$ with corresponding observed pixel set Ω^t, previous frame $\mathcal{Y}^{t-1}$ with corresponding Ω^{t-1}, previous tracked patch set $\mathcal{I}_\mathcal{S}$ with corresponding core tensors pool (set) $\mathcal{I}_\mathcal{Z}$, patch matching parameters b, m, o, s, l, K_b and K_o, thresholds τ_f and τ_c.

1: Pad the frames and get $\tilde{\mathcal{Y}}^{t-1}$ and $\tilde{\mathcal{Y}}^t$.
2: **for** each patch $\mathcal{S}_i$ in $\mathcal{I}_\mathcal{S}$ **do**
3:    Search K_o nearest patches within search range $l \times l$ in $\tilde{\mathcal{Y}}^t$.
4:    **if** Smallest distance $> \tau_f$ or overlap degree $> \tau_c$ **then**
5:       Remove $\mathcal{S}_i$ from $\mathcal{I}_\mathcal{S}$, remove corresponding core tensors from $\mathcal{I}_\mathcal{Z}$.
6:    **else**
7:       Replace $\mathcal{S}_i$ in $\mathcal{I}_\mathcal{S}$ with its first similar patch in $\tilde{\mathcal{Y}}^t$. Construct patch tensor $\mathcal{M}_i^t$ using the obtained K_o most similar patches. Update core tensors $\{\mathcal{Z}_{i,k}\}_{k=1}^N$ related to $\mathcal{S}_i$ using $\mathcal{M}_i^t$.
8:    **end if**
9: **end for**
10: Create new tracked patches set $\mathcal{I}'_\mathcal{S}$ to cover the uncovered region in $\tilde{\mathcal{Y}}_t$ according to $\mathcal{I}_\mathcal{S}$.
11: **for** each patch $\mathcal{S}_i$ in $\mathcal{I}'_\mathcal{S}$ **do**
12:    Search K_b nearest patches within search range $l \times l$ in $\tilde{\mathcal{Y}}_t$. Add $\mathcal{S}_i$ to $\mathcal{I}_\mathcal{S}$. Construct patch tensor $\mathcal{M}_i^t$ using obtained K_b nearest patches.
13:    Compute core tensors $\{\mathcal{Z}_{i,k}\}_{k=1}^N$ related to $\mathcal{S}_i$ using $\mathcal{M}_i^t$ and add them to $\mathcal{I}_\mathcal{Z}$.
14: **end for**
15: Obtain $\hat{\mathcal{Y}}^t$ by aggregating patches according to $\mathcal{I}_\mathcal{Z}$ and remove padded border pixels.
16: Compute recovered frame as $\mathcal{P}^t \circ \mathcal{Y}^t + (1 - \mathcal{P}^t) \circ \hat{\mathcal{Y}}^t$.
Output: Recovered frame of $\mathcal{Y}^t$, new tracked patch set $\mathcal{I}_\mathcal{S}$ with corresponding core tensors pool (set) $\mathcal{I}_\mathcal{Z}$.

IV. STREAMING TENSOR RING COMPLETION ALGORITHM

In this section, we develop an efficient streaming tensor completion algorithm for patch tensor completion. We adopt a tensor ring model to formulate the streaming tensor completion, which is verified to achieve better completion performance compared with the CP and Tucker model. To date, there is no streaming version for tensor ring completion. Further, we develop a new batch tensor ring completion algorithm to complete the new patch tensors and create the corresponding core tensors.

A. Tensor Ring Completion

First, let us revisit traditional tensor ring completion. Given a tensor $\mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and its observed index set Ω, the recovered full tensor $\mathcal{X}$ can be obtained by solving the following minimization problem

$$\min_{\mathcal{X}} \text{rank}_t(\mathcal{X}), \text{ s.t. } \mathcal{P} \circ \mathcal{X} = \mathcal{P} \circ \mathcal{M} \tag{7}$$

where $\text{rank}_t(\mathcal{X})$ is a tensor rank of $\mathcal{X}$.

According to Definition 1, the cost function for the tensor ring completion algorithm can be formulated as

$$\min_{\mathcal{Z}_1,\ldots,\mathcal{Z}_N} \|\mathcal{P} \circ (\mathcal{M} - \Re(\mathcal{Z}_1, \ldots, \mathcal{Z}_N))\|_F^2 . \tag{8}$$

The above problem can be solved using alternating least squares (ALS) [46]. Specifically, at each iteration, the core tensor $\mathcal{Z}_k$ is updated by fixing all core tensors except $\mathcal{Z}_k$, and based on Theorem 1, the following problem is solved

$$\min_{\mathcal{Z}_k} \left\| \mathbf{P}_{[k]} \circ \left(\mathbf{M}_{[k]} - \mathbf{Z}_{k(2)} \left(\mathbf{Z}_{[2]}^{\neq k}\right)^T\right) \right\|_F^2 . \tag{9}$$

The final estimated tensor $\mathcal{X}$ can be calculated by $\mathcal{X} = \Re(\mathcal{Z}_1, \ldots, \mathcal{Z}_N)$.

B. Streaming Tensor Ring Completion

Specific to the proposed patch-based streaming tensor ring completion framework, for a tracked patch in $\tilde{\mathcal{Y}}^{t-1}$, K_o nearest patches are found in $\tilde{\mathcal{Y}}^t$ and formed as a patch tensor $\mathcal{M}^t \in \mathbb{R}^{I_1 \times \ldots \times I_{N-1} \times K_o}$ with corresponding mask tensor $\mathcal{P}^t$. Further, it can be observed from (8) that only the last core tensor $\mathcal{Z}_N$ incorporates the temporal information of $\mathcal{M}$ (i.e., related to the temporal dimension). Thus, denoting by $\mathcal{V}^i \in \mathbb{R}^{r_N \times K_o \times r_1}$ the subtensor of $\mathcal{Z}_N$ related to $\mathcal{M}^i$, the tensor ring completion problem at time t using (8) can be rewritten as

$$\min_{\mathcal{Z}_1^t, \ldots, \mathcal{V}^t} \sum_{i=1}^t \left\| \mathcal{P}^i \circ \left(\mathcal{M}^i - \Re(\mathcal{Z}_1^t, \ldots, \mathcal{Z}_{N-1}^t, \mathcal{V}^i)\right) \right\|_F^2$$
$$+ \gamma \sum_{i=1}^t \|\mathcal{V}^i\|_F^2 \tag{10}$$

where we also included a regularization term to constrain the norm of each subtensor $\mathcal{V}^i$. γ is the regularization parameter.

For batch tensor ring completion (10) (or (8)), all tensors $\{\mathcal{M}^i\}_{i=1}^t$ and $\{\mathcal{P}^i\}_{i=1}^t$ must be accessible simultaneously for the completion process. However, as described in Algorithm 1, only the latest tensor $\mathcal{M}^t$ with the corresponding $\mathcal{P}^t$ and the previous core tensors $\{\mathcal{Z}_k^{t-1}\}_{k=1}^N$ are available at time t, which precludes the use of the current batch-based tensor ring completion methods. In this work, we develop a new streaming tensor ring completion algorithm to solve (10). The proposed method uses least-squares to get an estimate of $\mathcal{V}^t$ using $\{\mathcal{Z}_k^{t-1}\}_{k=1}^{N-1}$, followed by an approximation of $\{\mathcal{Z}_k^t\}_{k=1}^{N-1}$ using a scaled steepest descent (SSD) method.

1) Computation of $\mathcal{V}^t$: Suppose that at time t we have the tensor $\mathcal{M}^t$ and the previous estimated core tensors $\{\mathcal{Z}_k^{t-1}\}_{k=1}^{N-1}$. The optimization problem in terms of $\mathcal{V}^t$ in (10) becomes

$$\mathcal{V}^t = \arg\min_{\mathcal{V}} \left\| \mathbf{P}_{[N]}^t \circ (\mathbf{M}_{[N]}^t - \mathbf{V}_{(2)}(\mathbf{U}^{t-1})^T) \right\|_F^2 + \gamma \|\mathcal{V}\|_F^2 \tag{11}$$

where $\mathbf{U}^{t-1} = (\mathbf{Z}^{t-1})_{[2]}^{\neq N}$. Defining $\mathbf{v}_k \in \mathbb{R}^{r_N r_1}$ as the vectorized form of the k-th lateral slice of $\mathcal{V}^t$, the solution of (11) can be obtained by solving the following K_o subproblems:

$$\mathbf{v}_k = \arg\min_{\mathbf{v} \in \mathbb{R}^{r_N r_1}} \left\| \mathbf{p}_k \circ (\mathbf{m}_k - \mathbf{U}^{t-1}\mathbf{v}) \right\|_2^2 + \gamma \|\mathbf{v}\|_2^2, k = 1, \ldots, K_o \tag{12}$$

where $\mathbf{p}_k$ and $\mathbf{m}_k$ are the k-th rows of $\mathbf{P}_{[N]}^t$ and $\mathbf{M}_{[N]}^t$, respectively. Further, (12) can be simplified as

$$\mathbf{v}_k = \arg \min_{\mathbf{v} \in \mathbb{R}^{r_N r_1}} \left\| [\mathbf{m}_k]_{\mathcal{I}_k} - [\mathbf{U}^{t-1}]_{\mathcal{I}_k} \mathbf{v} \right\|_2^2 + \gamma \|\mathbf{v}\|_F^2 \quad (13)$$

where $\mathcal{I}_k$ denotes the index set of the non-zero entries of $\mathbf{p}_k$, and $[\mathbf{m}_k]_{\mathcal{I}_k} \in \mathbb{R}^{|\mathcal{I}_k|}$ denotes the vector of elements in $\mathbf{m}_k$ indexed by set $\mathcal{I}_k$ of cardinality $|\mathcal{I}_k|$. The matrix $[\mathbf{U}^{t-1}]_{\mathcal{I}_k} \in \mathbb{R}^{|\mathcal{I}_k| \times r_N r_1}$ is formed from the rows of $\mathbf{U}^{t-1}$ with row index $\mathcal{I}_k$. Eq. (13) has a closed-form solution

$$\mathbf{v}_k = \left([\mathbf{U}^{t-1}]_{\mathcal{I}_k}^T [\mathbf{U}^{t-1}]_{\mathcal{I}_k} + \gamma \mathbf{I} \right)^{-1} [\mathbf{U}^{t-1}]_{\mathcal{I}_k}^T [\mathbf{m}_k]_{\mathcal{I}_k}. \quad (14)$$

2) Update $\mathcal{Z}_1^t, \ldots, \mathcal{Z}_{N-1}^t$: Traditional streaming tensor completion methods usually apply stochastic gradient descent (SGD) or recursive least squares (RLS) to update the relative tensors. However, as shown in [29], directly applying SGD suffers from a low convergence rate, while the efficiency of RLS is limited due to its high computational complexity. In this work, we apply scaled steepest descent (SSD) method [47] to solve $\mathcal{Z}_1^t, \ldots, \mathcal{Z}_{N-1}^t$.

By only considering the data available at time t, we have $\mathcal{Z}_N^t = \mathcal{V}^t$ and the optimization problem in (10) at time t reduces to

$$\min_{\mathcal{Z}_1^t, \ldots, \mathcal{Z}_{N-1}^t} \mathcal{L} := \left\| \mathcal{P}^t \circ \left(\mathcal{M}^t - \Re(\mathcal{Z}_1^t, \ldots, \mathcal{Z}_{N-1}^t, \mathcal{Z}_N^t) \right) \right\|_F^2 \quad (15)$$

Traditional tensor ring completion algorithms used for solving (8) such as alternating least-squares (ALS) can be applied to solve (15). However, alternating minimization may incur a high computational cost, which is not computationally efficient for streaming processing. In this work, based on the fact that the tensor subspace within similar patches changes slowly (as Fig. 1 shows), we put forth a one-step SSD method to obtain the update of the core tensors.

First, $\mathcal{Z}_k^t, k = 1, \ldots, N-1$ are initialized with the corresponding previous estimates $\mathcal{Z}_k^{t-1}$. Then, the scaled gradient descent direction $g(\mathbf{Z}_{k(2)}^t)$ in terms of $\mathbf{Z}_{k(2)}^t$ is computed by first obtaining the gradient

$$\nabla \mathcal{L}(\mathbf{Z}_{k(2)}^t) = - \left(\mathbf{P}_{[k]}^t \circ (\mathbf{M}_{[k]}^t - \mathbf{Z}_{k(2)}^t \mathbf{U}_k^T) \right) \mathbf{U}_k \quad (16)$$

where $\mathbf{U}_k = (\mathbf{Z}^t)_{[2]}^{\neq k}$, and then adding a scaled term $\mathbf{U}_k^T \mathbf{U}_k$ such that

$$g(\mathbf{Z}_{k(2)}^t) = \nabla \mathcal{L}(\mathbf{Z}_{k(2)}^t)(\mathbf{U}_k^T \mathbf{U}_k + \epsilon \mathbf{I})^{-1}. \quad (17)$$

where ϵ is a sufficient small positive number (i.e., 10^{-10}). Therefore, $\mathbf{Z}_{k(2)}^t$ can be updated as

$$\mathbf{Z}_{k(2)}^t = \mathbf{Z}_{k(2)}^t - \mu_k g(\mathbf{Z}_{k(2)}^t), \quad (18)$$

where the step size μ_k is set by using exact line-search as

$$\mu_k = \arg \min_{\mu} \left\| \mathbf{P}_{[k]}^t \circ \left(\mathbf{M}_{[k]}^t - (\mathbf{Z}_{k(2)}^t - \mu g(\mathbf{Z}_{k(2)}^t)) \mathbf{U}_k^T \right) \right\|_F^2$$
$$= - \frac{\langle \nabla \mathcal{L}(\mathbf{Z}_{k(2)}^t), g(\mathbf{Z}_{k(2)}^t) \rangle}{\left\| \mathbf{P}_{[k]}^t \circ \left(g(\mathbf{Z}_{k(2)}^t) \mathbf{U}_k^T \right) \right\|_F^2}. \quad (19)$$

Compared with SGD, the SSD method increases the speed of convergence by using the scaled term $\mathbf{U}_k^T \mathbf{U}_k$ and applying an adaptive step size strategy. The above streaming tensor ring completion (STRC) algorithm is summarized in Algorithm 2.

Algorithm 2 Streaming Tensor Ring Completion (STRC)

Input: Partially observed tensor $\mathcal{M}^t$ with corresponding mask tensor $\mathcal{P}^t$ and previous estimation $\mathcal{Z}_k^{t-1}, k = 1, \ldots, N-1$. Parameter γ.
1: Compute $\mathcal{V}^t$ using (14) and set $\mathcal{Z}_N^t = \mathcal{V}^t$.
2: **for** $k = 1, \ldots, N-1$ **do**
3: 　Update $\mathcal{Z}_k^t$ using (18).
4: **end for**
Output: Updated core tensors $\mathcal{Z}_k^t, k = 1, \ldots, N$.

C. New Patch Tensor Rank Estimation and Completion

For a new patch tensor $\mathcal{M}^t \in \mathbb{R}^{I_1 \times \ldots \times I_{N-1} \times K_b}$ with corresponding mask tensor $\mathcal{P}^t$, there are no previous estimates of core tensors $\mathcal{Z}_k^{t-1}, k = 1, \ldots, N$. Therefore, one should apply a batch completion algorithm for $\mathcal{M}^t$ and create core tensors $\mathcal{Z}_k^t$. Inspired by the SSD method for STRC, we develop a new batch tensor ring completion algorithm using scaled steepest descent method (SSD). The method alternatively updates the core tensors $\mathcal{Z}_k^t, k = 1, \ldots, N$ using (18).

Before completion and before obtaining the core tensors of a new patch tensor, we should estimate the TR rank of the tensor. Following [25]–[27], [40], we set all the elements of the TR rank equal to the same value, i.e. $r_1 = \cdots = r_N = r$. Then, defining $p = \|\mathcal{P}^t\|_0 / (\prod_{i=1}^{N-1} I_i K_b)$ as the observation ratio of the tensor $\mathcal{M}^t$, according to [27, Theorem 1], we consider setting $r = C\sqrt{p}$, where C is a free parameter. In practice, if the patch is texture-less, the rank should be set to a relatively small value for better completion performance. Therefore, we propose the following adaptive strategy to estimate r:

$$r = \min(\lceil C_1 \sqrt{p} \sigma_{\mathcal{M}_\Omega}^2 \rceil, \lfloor C_2 \sqrt{p} \rfloor + r_o) \quad (20)$$

where $\sigma_{\mathcal{M}_\Omega}^2$ denotes the variance of the observed entries of $\mathcal{M}^t$, p the observation ratio of $\mathcal{M}^t$. C_1, C_2 and r_o are parameters to be set.

We term the method proposed above TRSSD, and the pseudo code is summarized in Algorithm 3. The method proposed can outperform least-square-based method for patch tensor completion, which is verified in the experimental results.

V. COMPLEXITY ANALYSIS

Without loss of generality, we assume that m and l are divisible by s, and the observation ratio for all patch tensors is the same value p. We first analyze the complexity of the proposed ECPM method. In particular, according to Section IV.B.2, the complexity of patch matching for each down-sampled patch is $\mathcal{O}(m^2 n l^2 / s^4)$. Therefore, the complexity of patch matching for a patch on all s^2 down-sampled frames is $\mathcal{O}(m^2 n l^2 / s^2)$. Suppose the total number of tracked patches for recovering the frame is T, then the overall complexity for ECPM on

Algorithm 3 Tensor Ring Completion With Rank Estimation and Scaled Steepest Descent (TRSSD)

Input: Partially observed tensor $\mathcal{M}$ with corresponding mask $\mathcal{P}$, maximum iteration number L and error tolerance ε.

1: Set $r_1 = \cdots = r_N = r$ and estimate r using (20).
2: Initialize $\mathcal{Z}_k, k = 1, \ldots, N$. Set $\tau = 1$.
3: **repeat**
4:   **for** $k = 1, \ldots, N$ **do**
5:     Update $\mathcal{Z}_k^\tau$ using (18).
6:     $\tau = \tau + 1$.
7:   **end for**
8:   Compute $\mathcal{X}^\tau = \Re(\mathcal{Z}_1^\tau, \mathcal{Z}_2^\tau, \ldots, \mathcal{Z}_N^\tau)$.
9: **until** $\tau > L$ or $\|\mathcal{X}^\tau - \mathcal{X}^{\tau-1}\|_F / \|\mathcal{X}^{\tau-1}\|_F < \varepsilon$

Output: Core tensors $\mathcal{Z}_k, k = 1, \ldots, N$.

Fig. 3. Left: curves of average PSNR with different interval s. Right: average time cost of patch tracking per frame.

all tracked patches is $\mathcal{O}(m^2 n l^2 / s^2 T)$. Compared with the traditional patch matching method (i.e., $s = 1$), the proposed ECPM method can reduce the computational complexity by a factor of s^2. Moreover, since both patch matching on T patches and coarse-scale matching for each patch on s^2 down-sampled frames are independent, the complexity can be further reduced by parallel computation.

Next we analyze the computational complexity of patch tensor completion. The time complexity of STRC and for a tracked patch tensor of size $m \times m \times n \times K_o$ is $\mathcal{O}(pm^2 n K_o r^4)$, while TRSSD for a new patch tensor of size $m \times m \times n \times K_b$ requires time complexity of $\mathcal{O}(pm^2 n K_b r^4 L)$. Define T_1 and T_2 as the number of tracked patch tensors and new patch tensors, respectively. Then, the overall complexity for tensor completion step is $\mathcal{O}(pm^2 n K_o r^4 T_1 + pm^2 n K_b r^4 L T_2)$. Moreover, since tensor completion for each patch tensor is independent, parallel computation can also be applied to further improve the computational efficiency.

VI. EXPERIMENTAL RESULTS

In this section, we conduct experiments to verify the performance of the proposed methods. We compare to both existing batch and streaming tensor completion algorithms. The streaming methods include the CP-based algorithm OLSTEC [29] and the t-SVD-based algorithm TOUCAN [31]. For batch methods, we compare to traditional algorithms including the matrix factorization-based algorithm TMAC [20], tensor ring-based algorithms TRNN [42] and PTRC [40], and t-SVD-based algorithms TNN [21] and TCTF [22]. We also compare two non-local (NL) patch-based batch methods including NLTNN [37] and tensor train-based NLTTNN [39].

The completion performance is evaluated by the average peak signal-to-noise ratio (PSNR) over 20 Monte Carlo runs with different missing entries. For all patch-based algorithms PTSTRC, NLTNN and NLTTNN, the patch size m and the number of overlap pixels o are set to 36 and 12, respectively. The number of nearest neighbors K for NLTNN and NLTTNN is set to 20, while K_b and K_o for PTSTRC are set to 30 and 10, respectively. The search size l for PTSTRC and NLTNN are both set to 41. The interval sampling parameter s, padding pixels b, threshold τ_f and τ_c for PTSTRC are set to 3,
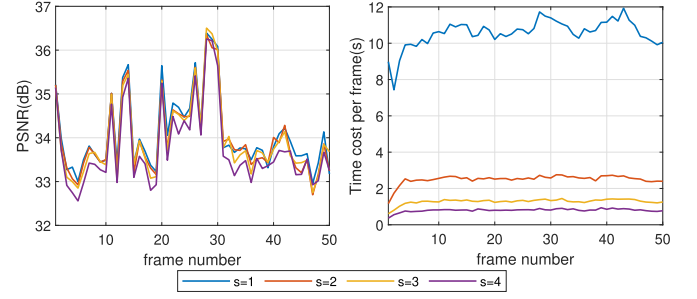
20, 0.02 and 3 respectively. For the proposed STRC and TRSSD, the parameters γ, L, C_1, C_2, r_o and ε are set to 10^{-5}, 10, 1000, 6, 4 and 10^{-2}, respectively. For the other algorithms, the parameters are adjusted so as to achieve their best performance. Further, the parameters are fixed in each simulation. All experiments were performed using MATLAB R2022a on a desktop PC with an 8-core 2.5-GHz processor i9-11900 and 32GB of RAM. Unless stated otherwise, parallel computation is applied to all patch-based methods for both patch tracking and tensor completion by using the MATLAB parallel computing toolbox on 8 CPU cores.

A. Ablation Experiment

To verify the effectiveness of the proposed patch tracking and tensor completion methods, in this section we carry out ablation experiments with different algorithm settings. The experiment is carried out on a 'water sport' video from the Moments in Time Dataset[1] [48]. Some representative frames are shown in the first row of Fig. 7. The first 50 frames of the video are selected, each frame with size $256 \times 342 \times 3$, resulting in a video tensor of size $256 \times 342 \times 3 \times 50$. The frames are assumed to arrive sequentially in a streaming fashion. The observation ratio is set to $p = 0.2$, i.e., the observed pixels are selected uniformly at random with probability 0.2. For fair comparison, parallel computation is not applied to patch tracking in this experiment.

First, we verify the efficiency of the proposed ECPM method. Fig. 3 depicts the PSNR of the recovered video for different values of interval parameter s, along with the computation time of patch matching for each frame. The dilation is utilized as default. Note that $s = 1$ designates patch matching on the full-scale. As can be seen, using the proposed ECPM method ($s > 1$) achieves significant speedup compared with full-scale matching. Further, as s increases, the matching accuracy decreases, underscoring a tradeoff between matching accuracy and efficiency.

Second, we evaluate the performance with different pre-processing methods for patch matching with missing data, i.e., dilation, interpolation and completion. Fig. 4 depicts the PSNR and patch matching time cost under different pre-processing methods. The prefix '+' and '−' mean 'with' and 'without', respectively. It should be noted that the result of
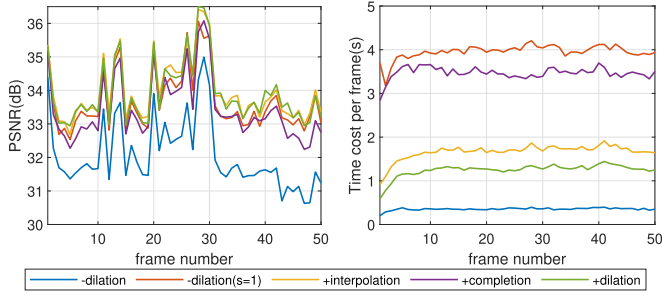
[1] http://moments.csail.mit.edu

Fig. 4. Left: curves of average PSNR with different pre-processing methods (*s* is set to 3 as default); Right: average time cost of patch tracking per frame.

Fig. 6. Left: curves of average PSNR with different thresholds τ_c and τ_f; Right: total number of patches for tensor completion per frame. Dotted blue line: results of completion using $\tau_c = 3$, $\tau_f = 0.02$ started from 51st frame.
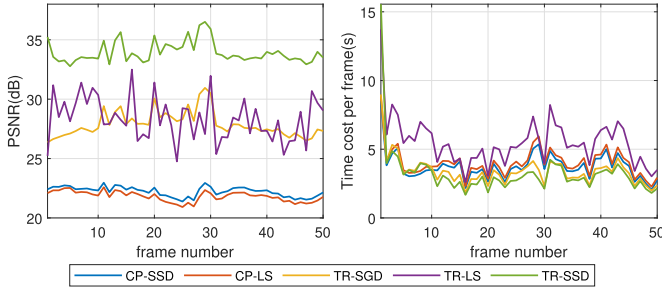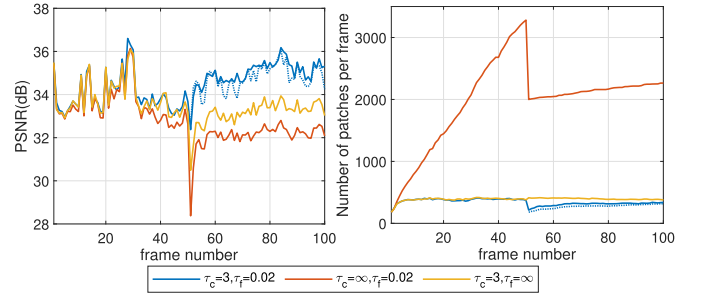

Fig. 5. Left: curves of average PSNR with different completion methods; Right: average time cost of tensor completion per frame.

$s = 1$ without dilation corresponds to the traditional patch tracking method [33]–[36]. One can observe that all methods achieve comparable good completion performance except for the one with $s = 3$ without dilation, while the proposed dilation method with $s = 3$ incurs a smaller computational cost for patch matching compared with interpolation and completion. Further, compared with Fig. 3, dilation greatly improves the patch matching accuracy, while increasing the computational cost. On the other hand, the speedup achieved due to coarse-scale patch matching counteracts the extra computational cost for matching using dilation. Therefore, our proposed patch tracking method can maintain both efficiency and accuracy.

Third, we test performance using different tensor ring completion methods. For comparison, we developed two extra streaming tensor ring completion methods TR-SGD and TR-LS. The TR-SGD method utilizes the gradient in (16) with exact line-search for updating $\{\mathcal{Z}_i\}_{i=1}^{N-1}$. The least-squares method TR-LS applies one-step LS to update $\{\mathcal{Z}_i\}_{i=1}^{N-1}$. To compare with the CP-based algorithm, we also develop two methods CP-SSD and CP-LS by introducing SSD and LS to the CP model, respectively. The PSNR and completion time cost with different completion methods are shown in Fig. 5. As shown, the proposed SSD method using the tensor ring model achieves the best completion performance, with computational cost smaller than that of TR-LS.

Finally, we verify the effectiveness of pruning mistracked or highly overlapping patches. The test video is constructed by concatenating two videos (D1 and S3 in Fig. 7), thus there exists a sudden scene change at the 51st frame. Similar to the previous experiments, the observed pixels are selected

uniformly at random with probability 0.2. The PSNR and the total number of patches (sum of tracked patches and new patches) for completion are shown in Fig. 6 for different parameter settings. The $\tau_c = \infty$ denotes the approach without pruning overlapping patches, while $\tau_f = \infty$ is without pruning mistracked patches. One can observe that pruning mistracked and highly overlapping patches significantly improves the completion performance when the scene changes. Specifically, pruning overlapping patches significantly suppresses the growth of the number of patches, while pruning mistracked patches greatly improves the completion performance and reduces the number of patches when the scene changes. For comparison, we also show the completion results on the second video S3 alone (dotted blue line in Fig. 6), i.e., the completion starts from the 51st frame. As shown, with the proposed pruning strategies, the completion performance is not affected when the scene changes.

B. Short Video Completion

In this part, we evaluate the completion performance using 50 short color videos from the Moments in Time Dataset [48]. The Moments in Time Dataset is a large-scale short video dataset initially developed for recognizing and understanding actions in videos. In our experiment, 50 videos from the dataset are randomly selected for performance test. Specifically, we randomly select 25 videos with dynamic camera (i.e., the camera moves during the shoot) and 25 videos with static camera (i.e., the camera does not move during the shoot). The dynamic and static camera videos are marked as D1-D25 and S1-S25, respectively. Representative frames from selected videos are shown in Fig. 7. In general, dynamic camera videos always contain large motion across the video frames.

Considering the computational limit of the batch tensor completion methods, for each video the first 50 frames are selected such that the whole tensor is of size $I_1 \times I_2 \times 3 \times 50$. If the height is larger than 256, the video is scaled down to a height of 256. Frames arrive sequentially for TOUCAN, OLSTEC and PTSTRC, while for batch methods, the whole video tensor with all 50 frames is given as the input tensor. For PTRC and TRNN (which favor high order tensors for better performance), we reshape the tensor to a 9/10/11-order tensor according to the factorization of I_1 and I_2 [40]. For

Fig. 7. Representative frames of six videos. Frame number (from left to right): 5, 15, 25, 35 and 45. Video indices (from top to bottom): D1, D5, D18, S3, S18 and S30.
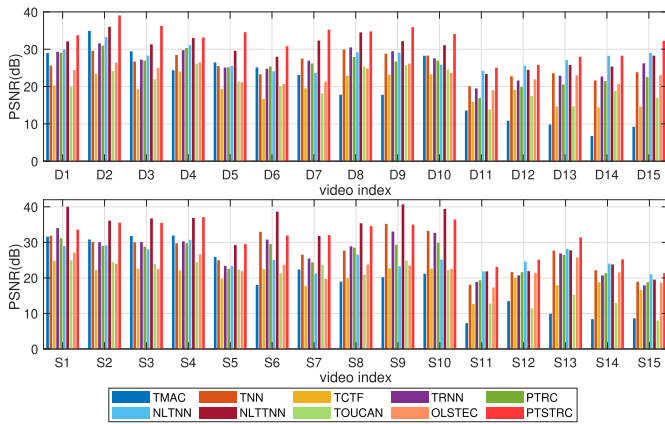

Fig. 8. Average PSNR for each video using different algorithms corresponding to Table I under $p = 0.2$. Top: dynamic camera videos D1-D15; Bottom: static camera videos S1-S15.

TNN and TCTF, which only afford 3-order tensor completion, we reshape the tensor to a 3-order tensor of size $I_1 \times I_2 \times 150$.

First, we investigate the performance of all algorithms on three missing data patterns commonly used in tensor completion performance verification, namely, random pixel/stripe/tube. In particular, in random pixel, the pixels are observed uniformly at random with observation ratio $p \times 100\%$. For the random stripe missing pattern, the observed columns and rows are randomly and uniformly selected with probability $p \times 100\%$. For random tube, a time-invariant random missing pattern with observation ratio $p \times 100\%$ is applied to each frame, i.e., the locations of the observed pixels are the same for all 50 frames.

Table I shows the average PSNR of different algorithms for dynamic and static camera videos. Each value of PSNR is averaged over a group of five videos. More detailed PSNR results for each video with $p = 0.2$ are shown in Fig. 8, and the corresponding average running times are shown in Fig. 9.
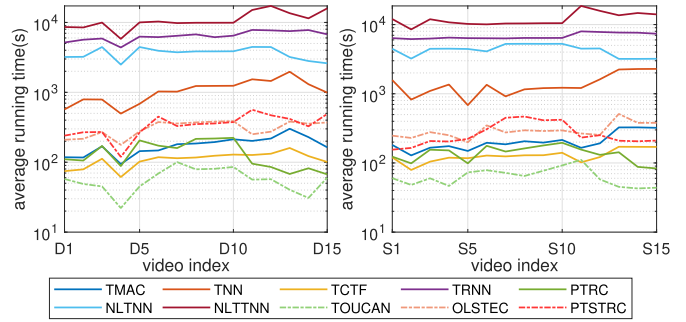

Fig. 9. Average running time for each video corresponding to Fig.8 ($p = 0.2$). Left: dynamic camera videos D1-D15; Right: static camera videos S1-S15.

For better visualization, examples of recovered frames using different algorithms are shown in Fig. 10. It can be observed that the proposed PTSTRC achieves the best performance for dynamic cases and second best for static cases. The superior performance of PTSTRC on dynamic videos confirms that the completion performance can be improved by leveraging temporal consistency within similar patches. Further, although NLTTNN outperforms PTSTRC on S1-S10, its computational cost is considerably larger than PTSTRC as shown in Fig. 9. More importantly, NLTTNN is a batch method that needs to access all 50 frames at one time, while our proposed PTSTRC is a streaming method that can sequentially complete each frame.

Second, we test the performance on two missing data patterns that are more common in practical applications. The first is the random block missing pattern, which could arise in video recovery with packet loss [49]. In particular, for each frame, a number $N \in [50, 150]$ of blocks with height $H \in [1, 10]$ and width $W \in [1, 100]$ are randomly generated and selected. In each block, partial color channel or all color channels are missing. The second is the watermark missing pattern which is directly related to the application of watermark removal. Specifically, in our experiment a fixed sentence is masked on all frames. Examples of some observed frames are shown in Fig. 11.

We also include two popular learning-based video inpainting methods, Spatial-Temporal Transformer Network (STTN) [50] and Video inpainting with Fuseformer (ViF) [51], for comparison. Table II shows the average PSNR for each video group for all algorithms, and examples of recovered frames for partial algorithms are shown in Fig. 11. TMAC, TCTF, TOUCAN and OLSTEC are not shown due to limited space. It can be observed that the proposed PTSTRC achieves the best overall performance with better visualization on dynamic videos. TNN or NLTTNN work better than PTSTRC on the random block missing pattern under static videos (S16-S20). An example of a recovered frame of S18 is shown in Fig. 11. It can be observed that the smearing of PTSTRC on missing regions is more obvious than TNN or NLTTNN.

C. Long Streaming Video Completion

In this section, we verify the completion performance on a live basketball video downloaded from YouTube.[2] We

[2] https://www.youtube.com/watch?v=I33o9UnUe1A&t=986s

TABLE I

AVERAGE PSNR ON SIX VIDEO GROUPS WITH DIFFERENT MISSING PATTERNS

| Camera Mode	Video index	Missing pattern	parameter p	batch methods							streaming methods		
				TMAC	TNN	TCTF	TRNN	PTRC	NLTNN	NLTTNN	TOUCAN	OLSTEC	PTSTRC
Dynamic	D1-D5	random pixel	0.1	20.72	23.89	19.67	24.71	25.51	26.93	28.39	20.46	21.66	**31.63**
			0.2	28.83	27.18	21.29	28.58	28.48	29.62	32.41	22.70	24.71	**35.35**
			0.3	32.02	29.39	22.11	31.83	30.16	31.66	35.57	24.39	26.46	**37.82**
	D6-D10	random stripe	0.1	12.33	24.03	16.60	23.09	18.08	22.64	25.77	20.41	19.28	**28.31**
			0.2	22.42	27.57	21.11	27.84	26.62	26.37	31.69	22.78	23.33	**34.15**
			0.3	26.40	30.69	22.66	31.95	30.02	29.15	35.56	24.60	26.21	**37.77**
	D11-D15	random tube	0.1	7.67	18.82	10.90	18.54	17.71	**24.05**	16.49	14.20	18.30	23.99
			0.2	10.04	22.37	15.76	22.58	20.26	26.84	25.44	16.38	21.55	**27.87**
			0.3	13.77	25.09	18.45	25.48	22.36	28.75	29.29	17.92	24.24	**30.41**
Static	S1-S5	random pixel	0.1	22.85	26.20	21.07	25.57	23.26	25.49	**31.80**	21.91	21.05	30.40
			0.2	30.41	29.36	22.31	29.57	28.25	28.05	**35.79**	24.02	24.41	34.24
			0.3	31.85	31.67	23.17	32.62	30.17	29.95	**38.79**	25.54	26.78	36.82
	S6-S10	random stripe	0.1	11.41	27.13	16.07	24.34	17.53	20.89	**28.85**	20.70	17.71	27.44
			0.2	20.14	31.12	21.08	30.16	28.32	24.25	**37.27**	22.58	22.64	34.01
			0.3	22.68	34.31	21.90	34.69	31.09	27.16	**41.03**	24.02	24.99	37.90
	S11-S15	random tube	0.1	7.55	18.30	11.13	17.70	18.84	21.70	14.51	9.52	18.02	**21.85**
			0.2	9.53	21.69	17.21	21.02	21.51	23.93	22.96	12.08	20.92	**25.24**
			0.3	12.11	24.05	20.28	23.54	23.38	25.55	26.92	8.26	22.91	**27.96**
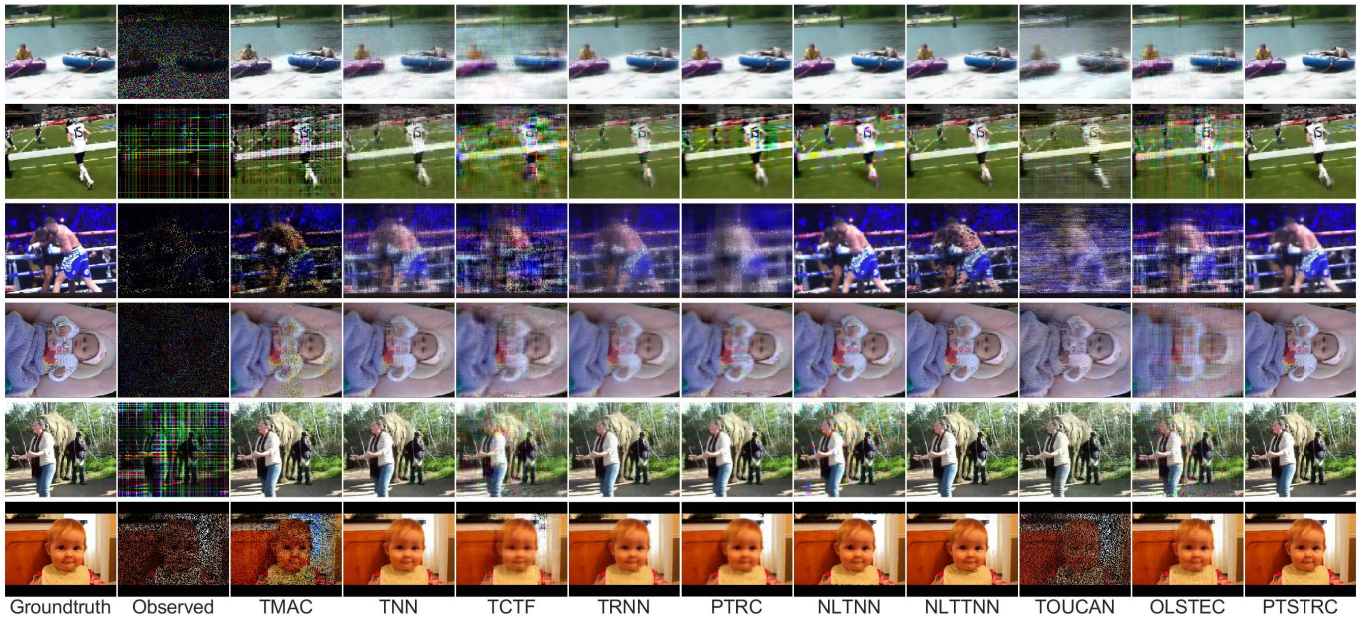
Fig. 10. Examples of recovered 25th frames of six videos corresponding to Table I. Video indices with p settings (from top to bottom): D1 ($p = 0.2$), D6 ($p = 0.1$), D11 ($p = 0.1$), S2 ($p = 0.1$), S7 ($p = 0.2$), S12 ($p = 0.3$). For better view, frames may be cropped on both sides.

select 500 consecutive frames and resize each frame to size 360×640. The frames arrive sequentially in a streaming manner. For each frame, the observed pixels are randomly and uniformly selected with probability 20%. Since batch completion methods cannot handle such a large tensor, we only compare the performance to the streaming completion methods OLSTEC and TOUCAN.

Fig. 12 depicts the PSNR of each recovered frame, along with the sum of magnitude of the optical flow between frames. Some sudden changes have been marked in the figure including two camera switch events in which the whole scene changes, and one flash overexposure where the frame is almost entirely white. Some recovered frames using different

algorithms are also displayed in Fig. 13. As shown, the proposed PTSTRC significantly outperforms OLSTEC and TOUCAN.

D. Image (Single Frame) Completion

In this part, we verify the performance of the proposed method on the task of image completion. As described in Remark 1, image completion can be regarded as a special case of streaming video completion where there is only one incoming frame at $t = 1$. In this case, only the TRSSD algorithm is utilized, and patch tensors are constructed using patch matching (PM) within the image. To avoid ambiguity, we name

TABLE II
AVERAGE PSNR ON FOUR VIDEO GROUPS WITH
DIFFERENT MISSING PATTERNS

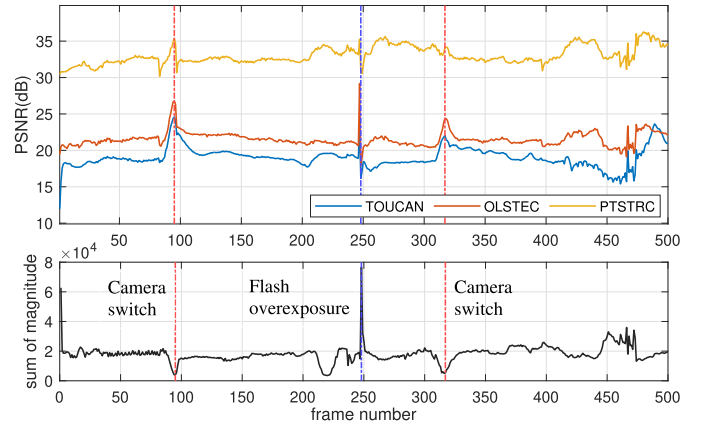| Camera mode		Dynamic		Static	
Video index		D16-D20	D21-D25	S16-S20	S21-S25
Missing pattern		random block	fixed watermark	random block	fixed watermark
batch methods	TMAC	19.40	19.68	18.91	18.38
	TNN	28.25	32.83	37.54	31.73
	TCTF	20.80	20.10	20.15	22.82
	TRNN	28.07	34.28	34.83	32.38
	PTRC	27.10	30.19	33.12	25.59
	NLTNN	27.65	36.11	27.90	34.25
	NLTTNN	30.31	36.98	**38.76**	34.76
learning methods	STTN	29.76	30.75	32.61	29.92
	ViF	30.47	32.21	32.19	29.99
streaming methods	TOUCAN	22.62	24.43	26.09	20.57
	OLSTEC	21.19	27.45	23.97	30.13
	PTSTRC	**30.71**	**38.59**	34.79	**35.28**

Fig. 12. Top: curves of average PSNR for each frame using different algorithms on basketball streaming video. Bottom: curves of average magnitude of optical flow between frames of the original fully observed video.
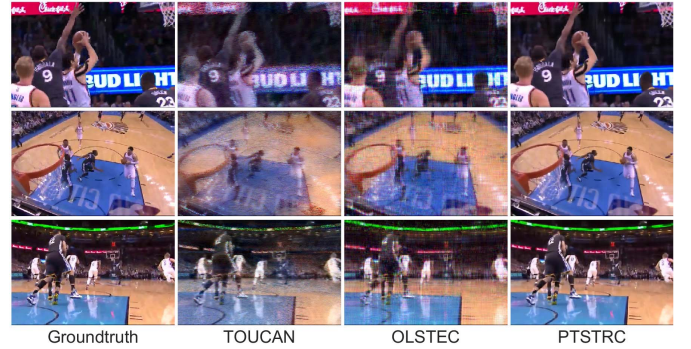

Fig. 13. Example of recovered frames (middle part). From top to bottom: frame 50, 200 and 450. Best viewed in $\times 2$ sized color pdf file.

Fig. 11. Examples of recovered 25th frames of three videos corresponding to Table II for partial algorithms. Video indices (from top to bottom): D19, D25 and S18. Bottom-right window of each frame shows the enlarged view of specific region (red rectangles in the groundtruth frames).

PTSTRC under this setting patch matching-based TRSSD (PMTRSSD). According to Remark 1, we also develop two new algorithms, namely, PMTTNN and PMTRNN, where TRNN [42] and TTNN [10] are utilized for patch tensor completion, respectively. The algorithm settings for PMTRSSD, PMTRNN and PMTTNN are the same as PTSTRC in the previous experiment.

We use the Berkeley Segment Dataset [52] as our test data; 30 images are randomly selected from the dataset and reshaped to a size of 320×480. For each image, $p \times 100\%$ of the pixels are randomly and uniformly selected as the observed data. The image completion problem is then formulated as a $320 \times 480 \times 3$ tensor completion task. For PTRC and TRNN,

we reshape the tensor to an 9-order tensor of size $4 \times 4 \times 4 \times 5 \times 4 \times 4 \times 5 \times 6 \times 3$. In this experiment, the number of nearest neighbors K for all patch-based method is set to 30.

First, we compare the completion performance of all algorithms on all the 30 test images. Fig. 14 shows the average PSNR for the test images with $p = 0.2$. As shown, our proposed method achieves the best performance and PMTRNN comes second. The results verify that the patch-based method can improve performance upon methods that treat the entire image as a tensor.

Second, we investigate the performance of the algorithms with different observation ratio p. Fig. 15 shows the completion performance for each algorithm for different values of p for the image 'bamboo house' (See Fig. 16). As can be seen, patch-based tensor ring methods (PMTRSSD and PMTRNN) achieve the overall best performance for all p compared with other methods. Moreover, for the low observability regime with $p < 0.3$, PMTRSSD outperforms PMTRNN in PSNR with a significantly smaller computational cost. When the number of observed pixels is relatively large (i.e., $p > 0.3$), PMTRNN has better performance than PMTRSSD. An example of recovered images under $p = 0.05$ is also provided in Fig. 16. As shown, the completed images using the proposed PMTRSSD and PMTRNN have better visual performance
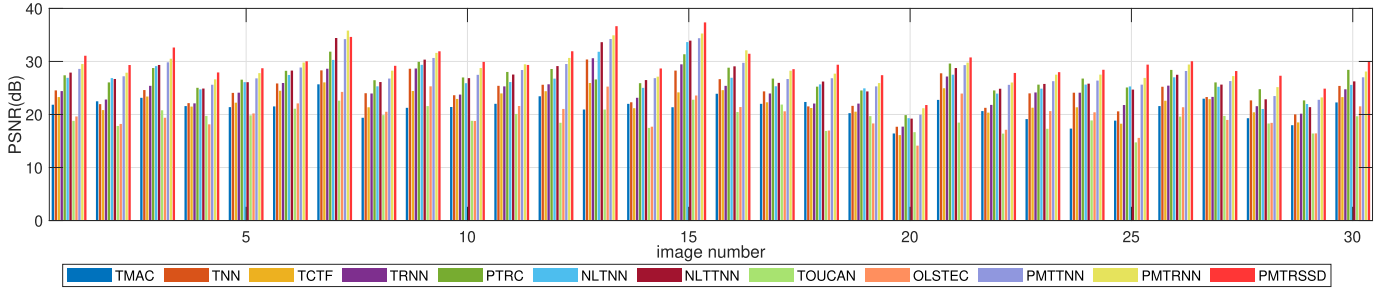
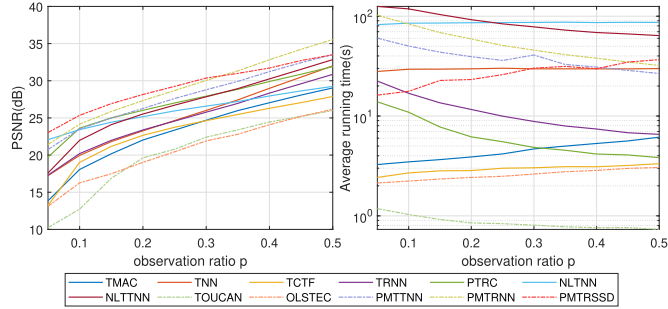Fig. 14. Comparison of the average PSNR for different algorithms on 30 selected test images with $p = 0.2$.

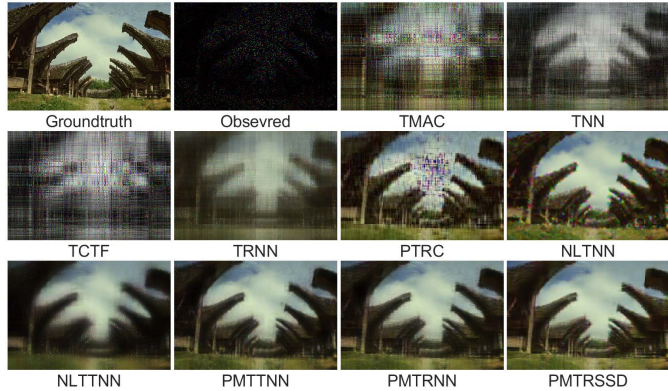Fig. 15. Left: average PSNR with different observation ratio p. Right: average time cost with different p.

Fig. 16. Example of recovered image in Fig. 15 with $p = 0.05$ using different algorithms. Best viewed in $\times 2$ sized color pdf file.

Fig. 17. Solid bar: average PSNR for the 25th frame of videos using different image completion algorithms. Dotted bar: the corresponding PSNR of the 25th frame from previous video completion results. The dotted bar of PMTRSSD denotes the PSNR of PTSTRC.

than other algorithms. It also shows that NLTNN suffers from color inconsistencies in some pixels due to misguidance from wrong interpolation. We should remark that although PMTRNN outperforms PMTRSSD in some cases, as mentioned in Remark 1, PMTRNN cannot be adapted to a streaming setting and its application is limited to image completion.

For streaming tensor completion where the frames arrive sequentially, a batch method can be applied on a frame-by-frame basis, i.e., each incoming frame $\mathcal{Y}^t$ can be treated as a single image and completed individually. We carry out an experiment to investigate the performance when completion is done frame-by-frame for the video data. In particular, each 25th frame of the 50 short videos used in the previous subsection is completed by image completion algorithms. The missing patterns of these frames are the same as in the previous
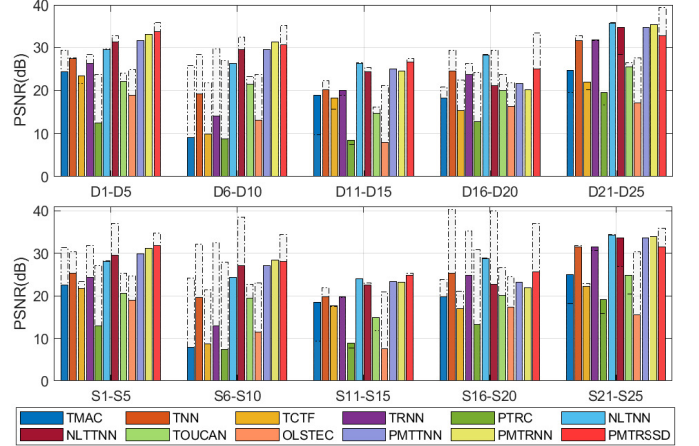
experiments ($p = 0.2$ for D1-D15 and S1-S15). Fig. 17 depicts the average PSNR for completion of the 10 groups corresponding to previous video groups. The dotted bars denote the corresponding average PSNR of the 25th frames obtained from the previous video completion results. As can be seen, most of the algorithms suffer from some performance degradation with image completion (solid bars) compared to the corresponding results using video completion (dotted bars) where the adjacent frames were considered together. This experiment verifies that it is beneficial to exploit the relation between frames rather than treat each frame individually.

VII. CONCLUSION

In this paper, we proposed a new patch tracking-based streaming tensor ring completion (PTSTRC) framework for visual data recovery. The framework exploits the high correlation between patches to improve the completion performance. To improve the patch matching efficiency, an efficient coarse-scale patch matching (ECPM) method is proposed. A new streaming tensor ring completion (STRC) algorithm using scaled steepest descent is developed, which can simultaneously achieve favorable completion performance and high computational efficiency. Equipped with an efficient ECPM method and a fast STRC algorithm, the proposed algorithm

can obtain better performance than the state-of-the-art batch and streaming tensor completion algorithms.

REFERENCES

[1] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: A tensor analysis," in *Proc. Conf. Learn. Theory*, 2016, pp. 698–728.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jan. 2017.

[3] K. Makantasis, A. D. Doulamis, N. D. Doulamis, and A. Nikitakis, "Tensor-based classification models for hyperspectral data analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 12, pp. 6884–6898, Dec. 2018.

[4] J. Kossaifi, Z. C. Lipton, A. Kolbeinsson, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," *J. Mach. Learn. Res.*, vol. 21, pp. 1–21, Jul. 2020.

[5] Y. Jia, H. Liu, J. Hou, S. Kwong, and Q. Zhang, "Multi-view spectral clustering tailored tensor low-rank representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4784–4797, Dec. 2021.

[6] M. Rupp and S. Schwarz, "A tensor LMS algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 3347–3351.

[7] M. Bousse, O. Debals, and L. De Lathauwer, "Tensor-based large-scale blind system identification using segmentation," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5770–5784, Nov. 2017.

[8] G. T. de Araujo, A. L. F. de Almeida, and R. Boyer, "Channel estimation for intelligent reflecting surface assisted MIMO systems: A tensor modeling approach," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 3, pp. 789–802, Apr. 2021.

[9] A. Sobral, S. Javed, S. K. Jung, T. Bouwmans, and E.-H. Zahzah, "Online stochastic tensor decomposition for background subtraction in multispectral video sequences," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 106–113.

[10] J. A. Bengua, H. N. Phien, H. D. Tuan, and M. N. Do, "Efficient tensor completion for color image and video recovery: Low-rank tensor train," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2466–2479, May 2017.

[11] T.-X. Jiang, T.-Z. Huang, X.-L. Zhao, L.-J. Deng, and Y. Wang, "A novel tensor-based video rain streaks removal approach via utilizing discriminatively intrinsic priors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4057–4066.

[12] W. Li, X. Zhu, and S. Gong, "Harmonious attention network for person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2285–2294.

[13] Y. Liu, Z. Long, H. Huang, and C. Zhu, "Low CP rank and tucker rank tensor completion for estimating missing components in image data," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 4, pp. 944–954, Apr. 2020.

[14] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, 2011, Art. no. 025010.

[15] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 208–220, Jan. 2013.

[16] A. H. Kiers, "Towards a standardized notation and terminology in multiway analysis," *J. Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.

[17] P. Jain and S. Oh, "Provable tensor factorization with missing data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1431–1439.

[18] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep. 1966.

[19] H. Kasai and B. Mishra, "Low-rank tensor completion: A Riemannian manifold preconditioning approach," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1012–1021.

[20] Y. Xu, R. Hao, W. Yin, and Z. Su, "Parallel matrix factorization for low-rank tensor completion," *Inverse Problems Imag.*, vol. 9, no. 2, pp. 601–624, 2015.

[21] Z. Zhang and S. Aeron, "Exact tensor completion using t-SVD," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1511–1526, Mar. 2016.

[22] P. Zhou, C. Lu, Z. Lin, and C. Zhang, "Tensor factorization for low-rank tensor completion," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1152–1163, Mar. 2017.

[23] X.-Y. Liu, S. Aeron, V. Aggarwal, and X. Wang, "Low-tubal-rank tensor completion using alternating minimization," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1714–1737, Mar. 2020.

[24] Q. Liu, X. Li, H. Cao, and Y. Wu, "From simulated to visual data: A robust low-rank tensor completion approach using l_p-regression for outlier resistance," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 6, pp. 3462–3474, Jun. 2022.

[25] W. Wang, V. Aggarwal, and S. Aeron, "Efficient low rank tensor ring completion," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5697–5705.

[26] L. Yuan, C. Li, D. Mandic, J. Cao, and Q. Zhao, "Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion," in *Proc. AAAI*, 2019, vol. 33, no. 1, pp. 9151–9158.

[27] H. Huang, Y. Liu, J. Liu, and C. Zhu, "Provable tensor ring completion," *Signal Process.*, vol. 171, Jun. 2020, Art. no. 107486.

[28] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Signal Process.*, vol. 63, no. 10, pp. 2663–2677, May 2015.

[29] H. Kasai, "Online low-rank tensor subspace tracking from incomplete data by CP decomposition using recursive least squares," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 2519–2523.

[30] Q. Song, X. Huang, H. Ge, J. Caverlee, and X. Hu, "Multi-aspect streaming tensor completion," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 435–443.

[31] K. Gilman and L. Balzano, "Online tensor completion and free submodule tracking with the T-SVD," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3282–3286.

[32] K. Xie *et al.*, "Accurate recovery of internet traffic data: A sequential tensor completion approach," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 793–806, Apr. 2018.

[33] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[34] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1791–1798.

[35] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 119–133, Apr. 2013.

[36] B. Wen, Y. Li, L. Pfister, and Y. Bresler, "Joint adaptive sparsity and low-rankness on the fly: An online tensor reconstruction scheme for video denoising," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 241–250.

[37] L. Zhang, L. Song, B. Du, and Y. Zhang, "Nonlocal low-rank tensor completion for visual data," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 673–685, Feb. 2021.

[38] M. K. Ng, X. Zhang, and X.-L. Zhao, "Patched-tube unitary transform for robust tensor completion," *Pattern Recognit.*, vol. 100, Apr. 2020, Art. no. 107181.

[39] M. Ding, T.-Z. Huang, X.-L. Zhao, M. K. Ng, and T.-H. Ma, "Tensor train rank minimization with nonlocal self-similarity for tensor completion," *Inverse Problem Imag.*, vol. 15, no. 3, pp. 475–498, 2021.

[40] J. Yu, G. Zhou, C. Li, Q. Zhao, and S. Xie, "Low tensor-ring rank completion by parallel matrix factorization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3020–3033, Jul. 2021.

[41] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor ring decomposition," 2016, *arXiv:1606.05535*.

[42] J. Yu, C. Li, Q. Zhao, and G. Zhao, "Tensor-ring nuclear norm minimization and application for visual: Data completion," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3142–3146.

[43] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, and L.-G. Chen, "Survey on block matching motion estimation algorithms and architectures with new results," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 42, no. 3, pp. 297–320, Mar. 2006.

[44] A. J. Hussain and Z. Ahmed, "A survey on video compression fast block matching algorithms," *Neurocomputing*, vol. 335, pp. 215–237, Mar. 2019.

[45] C. Je and H.-M. Park, "Optimized hierarchical block matching for fast and accurate image registration," *Signal Process., Image Commun.*, vol. 28, no. 7, pp. 779–791, Aug. 2013.

[46] L. Yuan, J. Cao, X. Zhao, Q. Wu, and Q. Zhao, "Higher-dimension tensor completion via low-rank tensor ring decomposition," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Nov. 2018, pp. 1071–1076.

[47] J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," *Appl. Comput. Harmon. Anal.*, vol. 40, no. 2, pp. 417–429, Mar. 2016.

[48] M. Monfort *et al.*, "Moments in time dataset: One million videos for event understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 502–508, Feb. 2020.

[49] N. Feamster and H. Balakrishnan, "Packet loss recovery for streaming video," in *Proc. 12th Int. Packet Video Workshop*, Pittsburgh, PA, USA, 2002, pp. 9–16.

[50] Y. Zeng, J. Fu, and H. Chao, "Learning joint spatial-temporal transformations for video inpainting," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 528–543.

[51] R. Liu *et al.*, "FuseFormer: Fusing fine-grained information in transformers for video inpainting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14040–14049.

[52] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Jun. 2001, pp. 416–423.

George K. Atia (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from Alexandria University, Egypt, in 2000 and 2003, respectively, and the Ph.D. degree from Boston University, MA, USA, in 2009, all in electrical and computer engineering. He joined the University of Central Florida in Fall 2012, where he is currently an Associate Professor with the Department of Electrical and Computer Engineering. He was a Visiting Faculty at the Air Force Research Laboratory (AFRL) from 2019 to 2020. From 2009 to 2012, he was a Post-Doctoral Research Associate at the Coordinated Science Laboratory (CSL), University of Illinois at Urbana-Champaign (UIUC). His research interests include machine learning and data analytics, explainable AI, statistical signal processing, detection and estimation theory, and information theory. He was a recipient of many awards, including the UCF Reach for the Stars Award and the CECS Research Excellence Award in 2018, the Dean's Advisory Board Fellowship and the Inaugural UCF Luminary Award in 2017, the NSF CAREER Award in 2016, and the Charles Millican Faculty Fellowship Award (2015–2017). He is an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.

Yicong He received the B.S. degree in automation and the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2012 and 2019, respectively. He is currently a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, University of Central Florida, FL, USA. His current research interests include robust signal processing and machine learning.