ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10097054

TENSOR LOW RANK COLUMN-WISE COMPRESSIVE SENSING FOR DYNAMIC IMAGING

Silpa Babu , Selin Aviyente** , Namrata Vaswani**

*Iowa State University, USA
**Michigan State University, USA

ABSTRACT

In recent work , we developed a fast, memory-efficient, and sample-efficient solution to the Low Rank column-wise Compressive Sensing (LRcCS) problem: recover an $n \times q$ LR matrix from m undersampled linear projections of each of its columns. Here, undersampled means $m \ll n, q$. The matrix LR model and the corresponding algorithms have two important limitations. First, for real image sequences, the required memory complexity is prohibitive. Secondly, for image or volume image sequences, it requires vectorizing the image or volume as one column of a matrix and this ignores the inherent 2D or 3D structure of the images or volumes. To address these limitations, in this work, we explore the use of a tensor LR model on the image sequence along with developing a fast and memory-efficient gradient descent (GD) based recovery algorithm and evaluating it experimentally.

1. INTRODUCTION

In recent work [1, 2], we developed a solution for the Low Rank column-wise Compressive Sensing (LRcCS) problem that is fast, memory-efficient, and sample-efficient when compared to other algorithms from the MRI literature as well as those from signal processing literature. The LRcCS problem is to recover an $n \times q$ LR matrix from m undersampled linear projections of each of its columns. Here, undersampled means $m \ll n, q$. This matrix LR model has two important limitations. First, for the recovery of Large-sized image sequences, consider any $n_1 \times n_2$ image, where $n = n_1 \cdot n_2$. For simplicity, suppose that $n_1 \geq n_2$. In these cases, the matrix version of our method (our older work) has time complexity that grows linearly with $n = n_1 \cdot n_2$. If n is very large, this is not fast enough.

Secondly, for image or volume image sequences, it requires vectorizing the image or volume as one column of a matrix and this ignores the inherent 2D or 3D structure of the images or volumes. To address these limitations, we explore the use of a tensor LR model on the image sequence along with developing a fast and memory-efficient gradient descent (GD) based recovery algorithm and evaluating it experimentally. Such a model has an obvious extension to volume imaging which will be explored in future work.

The above problems occur in undersampled LR based dynamic MRI [3, 2] and in federated sketching of long video sequences [4, 5]. Large scale usage of smartphones results in large amounts of geographically distributed data, e.g., images or videos. There is a need to compress/sketch this data before storing it. Sketch refers to a compression approach where the compression end is low complexity, usually random linear projections [4, 5, 1]. Sketching typically involves multiplying each vectorized image by a broad random matrix (matrix with fewer rows than columns), in order to compress it [4, 5, 1]. It can also involve left and right multiplying each image matrix by random matrices so that the resulting sketch is of lower dimensions than the original image. We use this latter model in the current work.

1.1. Existing work

CS theory for multidimensional data has become an emerging research area. Early work in the area attempts to find the best rank-R tensor approximation as a recovery of the original data tensor as in [6]. In [7], multi-way compressed sensing (MWCS) for sparse and low-rank tensors suggests a two-step recovery process: fitting a low-rank model in compressed domain, followed by per-mode decompression. However, the performance of MWCS relies highly on the estimation of the tensor rank, which is an NP-hard problem. [8] uses Kronecker product matrices in CS to act as sparsifying bases that jointly model the structure present in all of the signal dimensions as well as to represent the measurement protocols used in distributed settings. However, the recovery procedure, due to the vectorization of multidimensional signals, is rather time consuming and not applicable in practice. [9] propose Generalized Tensor Compressive Sensing (GTCS) with two reconstruction procedures, a serial method (GTCS-S) and a parallelizable method (GTCS-P). However, these methods assume that the underlying tensor is sparse and do not address the low-rank recovery problem.

One particular application where low-rank tensor recovery from compressed measurements has been encountered is dynamic MRI images. The problem of recovering dynamic MRI images from much smaller k-space measurements has been traditionally addressed by vectorizing each image in the sequence and enforcing low rank constraints on the resulting matrix. However, this vectorization leads to loss of inherent information such as spatial and temporal relationships and structures present in tensors. In [10, 11], the dynamic MRI data was reconstructed as a low-rank tensor, while in [12] the authors used a local low rank structure instead. Similarly, [13] proposes an accelerated imaging using Low-Rank Tensor with "Explicit Subspace" (LRTES) which represents high-dimensional image functions using an explicit low-rank tensor model. This model requires the acquisition of two complementary data sets: a navigator dataset for estimating the tensor subspace structure, and a sparse dataset for image reconstruction. More recently, low rank plus sparse assumption has been used to characterize dynamic MRI data [14].

1.2. Contribution

In this work, we assume a tensor LR model on the image sequence. This is specified in (2). We design an alternating gradient descent (GD) and minimization (altGDmin) algorithm for recovering this LR tensor from its sketches. Moreover, we enhance our basic tensor-altGDmin algorithm to also include the two key steps that significantly improved recovery performance for dynamic MRI in the matrix setting in our previous work [2]: initial mean image computation, and a final model error correction step. Our proposed solution for the tensor case method is motivated by our older work.

This work was partially supported by NSF grant CIF-2115200.

	Time	Memory	Communic.
Matrix (Sketch)	$n_1^2 m_1^2 r_1^2 q$	$n_1^2 \cdot m_1^2$	$n_1^2 r_1^2$
Tensor (Sketch)	$n_1 m_1^2 r_1 q$	$n_1 \cdot m_1$	$n_1 r_1$
Matrix (MRI)	$n_1^2 (\log n_1) r_1^2 q$	$\max(n_1^2 r_1^2, r_1^2 q)$	$n_1^2 r_1^2$
Tensor (MRI)	$n_1 (\log n_1) r_1 q$	$\max(n_1 r_1, r_1^2 q)$	$n_1 r_1$

Table 1: We compare the per iteration time, memory, and communication complexity of the matrix and tensor models. Assuming w.l.o.g. that $m_1 = \max(m_1, m_2)$, $n_1 = \max(n_1, n_2)$, $r_1 = \max(r_1, r_2)$ so that $O(n_1.n_2) = O(n_1^2)$ etc.

However, the tensor case models the images as being 2D low-rank which is a very different and usually more appropriate model for image-sequences/videos. The algorithm design for this case is not a straightforward extension but requires significant modifications in the initialization step and in the parameter setting approaches. Our tensor-altGDmin algorithm is significantly faster, and significantly more memory-efficient compared with its matrix counterpart. In a distributed setting, its communication complexity will also be much lower. We provide a comparison of its memory, communication, and time complexity per iteration in Table 1. This table provides the complexities for both the sketching and the MRI applications; in the latter case, the measurement matrices do not need to be explicitly stored, and one can use the fast fourier transform (FFT) algorithm, so all complexities are lower. However, still the tensor case is more efficient than the matrix one. As noted in our earlier works [1, 2], the matrix-based altGDmin algorithm already has the best speed and memory and communication complexities compared with existing approaches and hence our current tensor approach is significantly more efficient also than most other existing methods.

We use extensive experiments on real video sequences sketched using two types of random matrices to show the power of our developed tensor-altGDmin algorithm. We demonstrate that our algorithm (with *one fixed set of parameter settings*, no parameter tuning) can accurately recover many real video sequences from undersampled Gaussian and sub-exponential measurements.

2. PROPOSED DATA MODEL AND ALGORITHM

We give the data model first, then the algorithm.

2.1. Data model

We wish to recover a third-order tensor $\boldsymbol{\mathcal{X}}$ of size $n_1 \times n_2 \times q$ from the available third-order tensor measurements $\boldsymbol{\mathcal{Y}}$ of size $m_1 \times m_2 \times q$ that are obtained as follows. We use $\boldsymbol{X}_k$ to denote the k-th frontal slice of $\boldsymbol{\mathcal{X}}$, i.e. $\boldsymbol{X}_k : \boldsymbol{\mathcal{X}}(:,:,k)$. Similarly for $\boldsymbol{\mathcal{Y}}$. We have

$$\boldsymbol{Y}_k \quad = \quad \boldsymbol{X}_k \times_1 \boldsymbol{\Phi}_k \times_2 \boldsymbol{\Psi}_k, \ \text{ for all } \ k \in [q] \tag{1}$$

Here $\times_j$ denotes the j-mode product of a tensor [15]. In this particular setting, it can be understood more simply as $\boldsymbol{Y}_k = \boldsymbol{\Phi}_k \boldsymbol{X}_k \boldsymbol{\Psi}_k^\top$ for all $k \in [q]$. The measurement matrices $\boldsymbol{\Phi}_k$ are of size $m_1 \times n_1$ and $\boldsymbol{\Psi}_k$ of size $m_2 \times n_2$ with $m_1 < n_1$ and $m_2 < n_2$ respectively.

We assume the following 2D low rank (LR) model on $\boldsymbol{\mathcal{X}}$, with ranks r_1, r_2:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U} \times_2 \boldsymbol{V} \tag{2}$$

where $\boldsymbol{\mathcal{G}}$ is a $r_1 \times r_2 \times q$ core tensor, $\boldsymbol{U}$ is $n_1 \times r_1$, and $\boldsymbol{V}$ is $n_2 \times r_2$. This can also be rewritten also as $\boldsymbol{X}_k = \boldsymbol{U} \boldsymbol{G}_k \boldsymbol{V}^\top$ for all $k \in [q]$.

Algorithm 1 *altGDmin-Tensor*

1: **Input:** $\boldsymbol{\mathcal{Y}}, \boldsymbol{\Phi}, \boldsymbol{\Psi}, k \in [q]$
2: **Parameters:** GD step size η_1 and η_2, Number of iterations T, Set rank r_1 and r_2 as mentioned in Section 2.3
3: **Initialization:**
4: Define the init tensor $\hat{\boldsymbol{\mathcal{X}}}_0$ as follows: $(\hat{\boldsymbol{X}}_0)_k = \boldsymbol{\Phi}_k^\top \boldsymbol{Y}_k \boldsymbol{\Psi}_k, \ k \in [q]$
5: Set $\boldsymbol{U} \leftarrow$ top r_1 singular vectors of $(\hat{\boldsymbol{\mathcal{X}}}_0)_{(1)}$.
6: Set $\boldsymbol{V} \leftarrow$ top r_2 singular vectors of $(\hat{\boldsymbol{\mathcal{X}}}_0)_{(2)}$.
7: **for** $t = 1$ **to** T **do**
8: Update $\boldsymbol{\mathcal{G}}$: $\boldsymbol{G}_k = (\boldsymbol{\Phi}_k \boldsymbol{U})^\dagger \boldsymbol{Y}_k ((\boldsymbol{\Psi}_k \boldsymbol{V})^\dagger)^\top$, for all $k \in [q]$
9: Compute gradient w.r.t $\boldsymbol{U}$: $\nabla_{\boldsymbol{U}} \boldsymbol{L} = \sum_{k=1}^q \boldsymbol{\Phi}_k^\top (\boldsymbol{Y}_k - \boldsymbol{\Phi}_k \boldsymbol{U} \boldsymbol{G}_k \boldsymbol{V}^\top \boldsymbol{\Psi}_k^\top) \boldsymbol{\Psi}_k \boldsymbol{V} \boldsymbol{G}_k^\top$
10: Compute gradient w.r.t $\boldsymbol{V}$: $\nabla_{\boldsymbol{V}} \boldsymbol{L} = \sum_{k=1}^q \boldsymbol{\Psi}_k^\top (\boldsymbol{Y}_k - \boldsymbol{\Phi}_k \boldsymbol{U} \boldsymbol{G}_k \boldsymbol{V}^\top \boldsymbol{\Psi}_k^\top)^\top \boldsymbol{\Phi}_k \boldsymbol{U} \boldsymbol{G}_k$
11: Update $\boldsymbol{U}, \boldsymbol{V}$ by projected GD: Let $QR(\boldsymbol{M})$ be the Q of QR decomp of $\boldsymbol{M}$. Set $\boldsymbol{U} \leftarrow QR(\boldsymbol{U} - \eta_1 \nabla_{\boldsymbol{U}} \boldsymbol{L})$ Set $\boldsymbol{V} \leftarrow QR(\boldsymbol{V} - \eta_2 \nabla_{\boldsymbol{V}} \boldsymbol{L})$
12: **end for**
13: **Output:** $\boldsymbol{\mathcal{X}}$ where $\boldsymbol{\mathcal{X}}(:,:,k) = \boldsymbol{X}_k = \boldsymbol{U} \boldsymbol{G}_k \boldsymbol{V}^\top$

2.2. The Tensor altGDmin algorithm

Using the 2D LR model specified above, our goal is to minimize the cost function

$$\boldsymbol{L}(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\mathcal{G}}) := \sum_{k=1}^{q} \| \boldsymbol{Y}_k - \boldsymbol{\Phi}_k \boldsymbol{U} \boldsymbol{G}_k \boldsymbol{V}^\top \boldsymbol{\Psi}_k^\top \|_F^2$$

over $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\mathcal{G}}$. We use the alternating GD and minimization (altGDmin) approach to do this, since it provides the best speed and memory/communication complexity tradeoff. By modifying ideas from matrix LR literature, we start with a carefully designed spectral initialization to initialize $\boldsymbol{U}$ and $\boldsymbol{V}$. After this we alternatively update $\boldsymbol{\mathcal{G}}$ and $\{\boldsymbol{U}, \boldsymbol{V}\}$ as follows.

1. Given $\{\boldsymbol{U}, \boldsymbol{V}\}$, update $\boldsymbol{\mathcal{G}}$ by minimizing the above cost function over it. Due to the form of our cost function, this decouples into a matrix-wise minimization for each $r_1 \times r_2$ frontal slice matrix $\boldsymbol{G}_k$ individually. Under the LR assumption, r_1, r_2 are small, and thus this step consists of q very inexpensive least squares (LS) problems. These can be solved in closed form. See line 8 of Algorithm 1.
2. Given $\boldsymbol{\mathcal{G}}$, we update $\{\boldsymbol{U}, \boldsymbol{V}\}$ using projected GD: one GD step, followed by projecting the output onto the space of matrices with orthonormal columns. This is done using the QR decomposition. This projection helps ensure that the norms of both $\{\boldsymbol{U}, \boldsymbol{V}\}$ and of $\boldsymbol{\mathcal{G}}$ remain bounded: the former remain unit norm. Without this projection step, the norm of one of $U, V, \boldsymbol{\mathcal{G}}$ could keep increasing while that of the other keeps decreasing. The QR decomposition is very cheap, it is of order $n_1 r_1^2$ and $n_2 r_2^2$ respectively.

For spectral initialization, we use a modification of our matrix case idea. We first define the initialization tensor $\hat{\boldsymbol{\mathcal{X}}}_0$ as follows: we set its k-th frontal slices as $(\hat{\boldsymbol{X}}_0)_k = \boldsymbol{\Phi}_k^\top \boldsymbol{Y}_k \boldsymbol{\Psi}_k, \ k \in [q]$. We initialize $\boldsymbol{U}$ as the top r_1 singular vectors of the unfolded matrix $(\boldsymbol{\mathcal{X}}_0)_{(1)}$: this means unfold $(\boldsymbol{\mathcal{X}}_0)$ along the first dimension to get a matrix of size $n_1 \times n_2 q$. Similarly, we initialize $\boldsymbol{V}$ as the top r_2 singular vectors

of the unfolded matrix $(\boldsymbol{\mathcal{X}}_0)_{(2)}$: this means unfold $(\boldsymbol{\mathcal{X}}_0)$ along the second dimension to get a matrix of size $n_2 \times n_1 q$.

The complete algorithm is specified in Algorithm 1

Time, Memory, and Communication complexity. For computing the complexities, assume that $n_1 = \max(n_1, n_2)$, $m_1 = \max(m_1, m_2)$, and $r_1 = \max(r_1, r_2)$ so that $O(n_1 \cdot n_2) = O(n_1^2)$, etc. With simple calculations, it is easy to see that the time complexity per iteration is $O((n_1 m_1 m_2 r_1 + n_2 m_1 m_2 r_2)q) = O(n_1 m_1^2 r_1 q)$. The memory complexity is governed by the memory needed to store the measurement matrices. Thus it is $O(n_1 \cdot m_1)$. For communication complexity, notice that in a distributed setting, the updates of $\boldsymbol{G}_k$s will be done locally while the gradients will need to be shared across the different nodes. The gradients are matrices of size $n_1 \times r_1$ and $n_2 \times r_2$. Thus the communication complexity is $O(n_1 r_1)$. We summarize this information in Table 1. In the MRI setting, due to use of the FFT operator, one only needs to store the sampled indices (which is much cheaper) and the time needed is also much lesser in both cases. However, still, the tensor case time and memory requirements are much lower. These are also given in the table.

Algorithm 2 *altGDmin-Tensor-Video*

Input: $\boldsymbol{\mathcal{Y}}, \boldsymbol{\Phi}_k, \boldsymbol{\Psi}_k, k \in [q]$

Parameters: GD step size $\eta = \frac{1}{7\|\nabla_U f(\boldsymbol{UB})\|}$, Number of iterations $T = 70$, $\epsilon_{exit} = 0.01$

1. **Mean Image estimate:**
   (a) Solve $\min_{\bar{\boldsymbol{Z}}} \sum_k ||\boldsymbol{Y}_k - \boldsymbol{\Phi}_k \bar{\boldsymbol{Z}} \boldsymbol{\Psi}_k^\top||_F^2$; denote solution by $\hat{\bar{\boldsymbol{Z}}}$.
2. **AltGDmin on measurements' residual:**
   (a) Compute $\tilde{\boldsymbol{Y}}_k := \boldsymbol{Y}_k - \boldsymbol{\Phi}_k \bar{\boldsymbol{Z}} \boldsymbol{\Psi}_k^\top$ for all $k \in [q]$.
   (b) Give $\tilde{\boldsymbol{Y}}_k, \boldsymbol{\Phi}_k, \boldsymbol{\Psi}_k, k \in [q], r_1, r_2$ as the input of Algorithm 1 and run it. Denote its output by $\hat{\boldsymbol{\mathcal{X}}}$.
3. **MEC on measurements' residual:** For each $k \in q$,
   (a) compute $\tilde{\tilde{\boldsymbol{Y}}}_k := \boldsymbol{Y}_k - \boldsymbol{\Phi}_k \bar{\boldsymbol{Z}} \boldsymbol{\Psi}_k^\top - \boldsymbol{\Phi}_k \hat{\boldsymbol{X}}_k \boldsymbol{\Psi}_k^\top$
   (b) run 3 iterations of GD to solve $\min_{\boldsymbol{E}} \|\tilde{\tilde{\boldsymbol{Y}}}_k - \boldsymbol{\Phi}_k \boldsymbol{E}_k \boldsymbol{\Psi}_k^\top\|_F^2$. Denote the output by $\hat{\boldsymbol{\mathcal{E}}}$.

Output : $\hat{\boldsymbol{\mathcal{Z}}}$ with $\hat{\boldsymbol{\mathcal{Z}}}(:,:,k) = \hat{\boldsymbol{Z}}_k = \hat{\bar{\boldsymbol{Z}}} + \hat{\boldsymbol{X}}_k + \hat{\boldsymbol{E}}_k$.

3. TENSOR-ALTGDMIN FOR REAL VIDEOS

We enhance our basic tensor altGDmin with two extra steps explained below. Next we explain how we set parameters. These are set once and used for *all* our experiments.

3.1. Realistic image modeling and algorithm

Most real video sequences have a certain baseline component that is roughly constant across the entire sequence. We refer to this as the "mean" image. Secondly, most real image sequences are only approximately LR, i.e., the residual after subtracting the LR component is not zero, but has a small magnitude. Thus the following is a more appropriate model for a real image sequence:

$$\boldsymbol{\mathcal{Z}}(:,:,k) = \boldsymbol{Z}_k = \bar{\boldsymbol{Z}} + \boldsymbol{X}_k + \boldsymbol{E}_k, \quad \text{for all } \; k \in [q]$$

where $\boldsymbol{\mathcal{Z}}(:,:,k)$ is the k-th image in the image sequence, $\bar{\boldsymbol{Z}}$ is the mean image (this is typically the background image), the $\boldsymbol{X}_k$'s forms a 2D LR tensor $\boldsymbol{\mathcal{X}}$ (this is the slow moving component), $\boldsymbol{E}_k$ is the modeling error in this modeling. We also assume $||\bar{\boldsymbol{Z}}||_F \gg ||\boldsymbol{X}_k||_F \gg ||\boldsymbol{E}_k||_F$ for all k. This is modeled as being unstructured and small magnitude. We recover the mean image using an LS step that involves all measurements. See step 1 of Algorithm 2. We use the mean estimate to compute the measurement residuals as in step 2a of the algorithm and use these as input the tensor-altGDmin algorithm (step 2b). Finally, we recover the model errors $\boldsymbol{E}_k$ for each k individually by running a few steps of GD on the new measurement residuals (see step 3a and 3b). We refer to this step as model error correction (MEC).

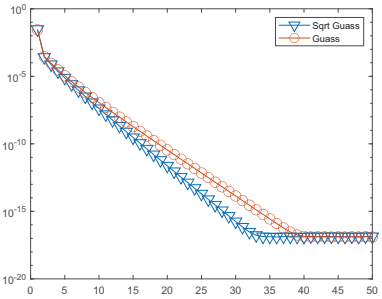

Fig. 1: *Error versus number of iterations for simulated data.*

3.2. Setting parameters automatically

One way to choose r_1 and r_2 is to use the "$b\%$ energy threshold" on the singular values of the initialization matrices $\boldsymbol{\mathcal{X}}_{(1)}$ and $\boldsymbol{\mathcal{X}}_{(2)}$. In addition, r_1 needs to be sufficiently small compared $\min(n_1, n_2 q)$ for the algorithm to take advantage of the LR assumption. Similarly $r_2 \ll \min(n_1 q, n_2)$. Also, for the LS step to update $\boldsymbol{G}_k$'s to work well, we need $r_1 \ll m_1$ and $r_2 \ll m_2$. Thus, we let $r_{1,big} = \min(n_1, n_2, q, m_1)/20$ and $r_{2,big} = \min(n_1, n_2, q, m_2)/20$. We then compute r_1 as the smallest integer so that the squared sum of the first r_1 singular values of $\boldsymbol{\mathcal{X}}_{(1)}$ is at least 85% of the squared sum of the first $r_{1,big}$. Similarly for r_2.

Using ideas from [2], we choose the GD step size as $\eta_1 = \frac{1}{5\|\nabla_{\boldsymbol{U}}(\boldsymbol{L}(\boldsymbol{U}^0, \boldsymbol{\mathcal{G}}^0, \boldsymbol{V}^0))\|}$, $\eta_2 = \frac{1}{5\|\nabla_{\boldsymbol{V}}(\boldsymbol{L}(\boldsymbol{U}^0, \boldsymbol{\mathcal{G}}^0, \boldsymbol{V}^0))\|}$ where the $\boldsymbol{U}^0, \boldsymbol{V}^0, \boldsymbol{\mathcal{G}}^0$ are the initial estimates. The maximum number of iterations, T, is set to 70 and also we stop the GD loop when $Error(\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{X}}) < 0.001$. For mean image computation, we use the CGLS code `https://web.stanford.edu/group/SOL/software/cgls/` with tolerance 10^{-6} and with maximum 10 iterations. For the MEC step, we run at most 10 iterations of CGLS code with tolerance 10^{-36}.

4. EXPERIMENTS

In all our experiments, Error is computed as $Error(\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{X}}) = \frac{\|(\boldsymbol{\mathcal{X}}^*)_{vec} - (\boldsymbol{\mathcal{X}})_{vec}\|}{\|(\boldsymbol{\mathcal{X}}^*)_{vec}\|}$. We experiment with two types of sketching matrices $\boldsymbol{\Phi}_k, \boldsymbol{\Psi}_k$: both are random Gaussian (each entry is i.i.d. standard Gaussian) or both are what we call "square-root-Gaussian (sqrtG)". Each entry of a sqrtG matrix is the square root of the absolute value of a standard Gaussian random variable (r.v.) multiplied by the sign of this standard Gaussian r.v.. In the sqrtG case, it can be argued that each entry of the sketched/measurement tensor $\boldsymbol{\mathcal{Y}}$ is an independent sub-Gaussian r.v., while in the Gaussian case, each entry of $\boldsymbol{\mathcal{Y}}$ is an independent sub-exponential r.v.. The latter is known to be heavier tailed than the former and hence one would expect slower algorithm convergence or larger final error or both. In our first experiment, we simulated an exactly 2D-LR image sequence and its measurements using our data model. We generated

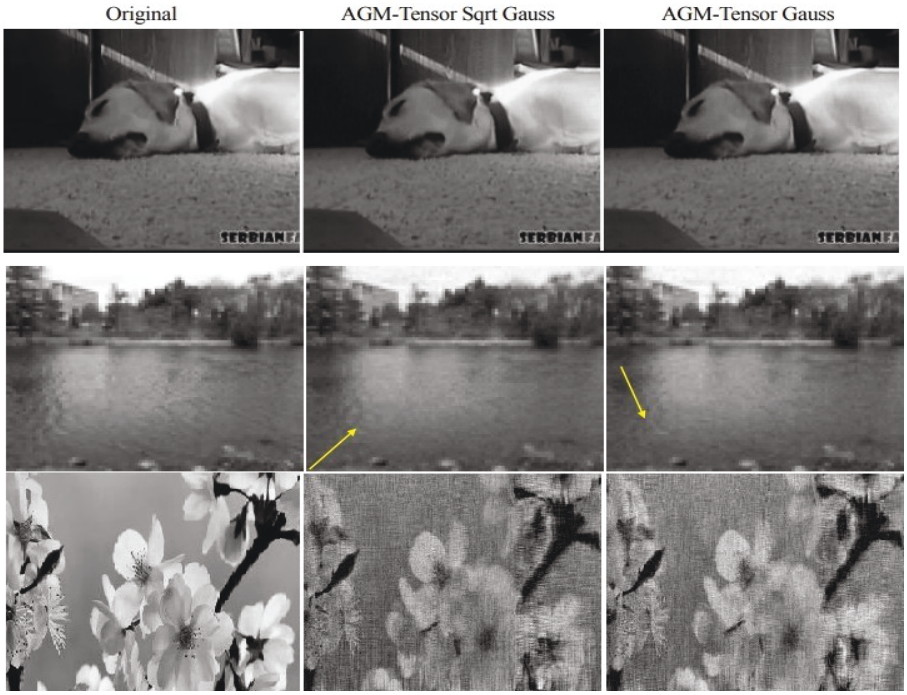

Fig. 2: *Visual image quality comparison of the reconstruction using AGM-Tensor of Lake (Frame* 70*), Sleeping Dog (Frame* 70*) and Blooms (Frame* 1*) Video. The yellow arrow shows the blurring*

$\boldsymbol{U}^*$ and $\boldsymbol{V}^*$ by orthogonalizing standard Guassian matrix of dimension $n_1 \times r_1$ and $n_2 \times r_2$ respectively while $\boldsymbol{\mathcal{G}}^*$ is a tensor of size $r_1 \times r_2 \times q$ with each entry being i.i.d. standard Gaussian. Since sub-exponential r.v.s are heavier tailed, the algorithm takes longer to converge in the latter case. We show the plot in Fig 1. This used $n_1 = 70$, $n_2 = 200$, $q = 300$, $r_1 = 3$, $r_2 = 4$, $m_1 = 60$ and $m_2 = 120$.

Next, we experiment with sketching of the following six different real video sequences (only approximately LR) which are different in terms of size and motion using sqrtG and Gaussian sketches: Switch Light (n_1=120, n_2=160, q=770), Dog Sleeping (n_1=180, n_2=320, q=148), Lake (n_1=72, n_2=90, q=200), Escalator (n_1=130, n_2=160, q=200), Person (n_1=120, n_2=160, q=200), Lobby(n_1=128, n_2=160, q=341). These were taken from `https://github.com/praneethmurthy/ReProCS/tree/master/Data` which obtained them from `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html` (this link does not work now), `https://pixabay.com/videos/blooms-tree-blossoms-spring-branch-113004/` was used to get the bloom (n_1=300, n_2=500, q=600) video. We used $m_1/n_1 = m_2/n_2$ ranging from 0.4 to 0.9. In each row we specify this ratio. We sketched these using both sqrtG and Gaussian matrices. We display the recovery errors *and SSIM* for all 14 cases in Table 2. We show one recovered frame of three of these videos in Fig. 2. As can be seen, the recovered image quality is worse when using Gaussian sketches.

Dataset		Sqrt Gauss	Gauss
Lake, $m_1 = 0.4n_1$	Error	0.0565	0.0565
	SSIM	0.6770	0.6718
Dog, $m_1 = 0.6n_1$	Error	0.0241	0.0241
	SSIM	0.9183	0.9206
SwitchLight, $m_1 = 0.9n_1$	Error	0.0088	0.0117
	SSIM	0.9883	0.9886
Lake, $m_1 = 0.9n_1$	Error	0.0364	0.0366
	SSIM	0.8237	0.8146
Escalator, $m_1 = 0.9n_1$	Error	0.0847	0.0850
	SSIM	0.7862	0.7924
Person, $m_1 = 0.9n_1$	Error	0.1382	0.1386
	SSIM	0.5421	0.5459
Lobby, $m_1 = 0.9n_1$	Error	0.0144	0.0143
	SSIM	0.9240	0.9200
Blooms, $m_1 = 0.9n_1$	Error	0.0950	0.0951
	SSIM	0.5149	0.5166

Table 2: *Recovery Error and SSIM results for 7 different image sequences using both square-root Gaussian and Gaussian measurements. The ratio $m_2/n_2 = m_1/n_1$ in all cases. SSIM is a measure of correlation, so higher is better.*

Finally, we would like to compare with our old altGDmin-matrix algorithm [2], but we are unable to do this, due to the large memory requirement. The matrix case would use a measurement matrix $\boldsymbol{A}_k = \boldsymbol{\Phi}_k \otimes \Psi_k^\top$. This is of size $m_1 m_2 \times n_1 n_2$. As an example for the Dog-Sleeping sequence, if $m_1/n_1 = m_2/n_2 = 0.6$, the memory required to store just one such $\boldsymbol{A}_k$ is more than one gigabyte. The time needed by one iteration of matrix altGDmin is 3.6 seconds while the tensor version needs 0.06 seconds. The final errors are 0.09 (matrix) and 0.13 (tensor).

5. CONCLUSIONS

We explored the use of a tensor LR model for sketching long and large-sized image sequences. We developed a very fast and memory-efficient gradient descent (GD) based recovery algorithm, called altGDmin-tensor and showed that it can successfully recover both simulated and real video image sequences from their sketches. Such a model has an obvious extension to dynamic volume imaging which is routinely done in dynamic MRI. In future work we will develop and evaluate our approach for this application. In MRI, approaches that enable accurate reconstructions from highly undersampled dynamic volume image sequences can be very useful.

6. REFERENCES

[1] S. Nayer and N. Vaswani, "Fast and sample-efficient federated low rank matrix recovery from column-wise linear and quadratic projections," *IEEE Trans. Info. Th.*, 2022, to appear. Also at arXiv:2102.10217.

[2] S. Babu, S. Nayer, S. G. Lingala, and N. Vaswani, "Fast low rank compressive sensing for accelerated dynamic mri," in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, 2022.

[3] S. G. Lingala, Y. Hu, E. DiBella, and M. Jacob, "Accelerated dynamic mri exploiting sparsity and low-rank structure: kt slr," *IEEE Transactions on Medical Imaging*, vol. 30, no. 5, pp. 1042–1054, 2011.

[4] F. P. Anaraki and S. Hughes, "Memory and computation efficient pca via very sparse random projections," in *Intl. Conf. Machine Learning (ICML)*, 2014, pp. 1341–1349.

[5] R. S. Srinivasa, K. Lee, M. Junge, and J. Romberg, "Decentralized sketching of low rank matrices," in *Neur. Info. Proc. Sys. (NeurIPS)*, 2019, pp. 10 101–10 110.

[6] L.-H. Lim and P. Comon, "Multiarray signal processing: Tensor decomposition meets compressed sensing," *Comptes Rendus Mecanique*, vol. 338, no. 6, pp. 311–320, 2010.

[7] N. D. Sidiropoulos and A. Kyrillidis, "Multi-way compressed sensing for sparse low-rank tensors," *IEEE Signal Processing Letters*, vol. 19, no. 11, pp. 757–760, 2012.

[8] M. F. Duarte and R. G. Baraniuk, "Kronecker compressive sensing," *IEEE Transactions on Image Processing*, vol. 21, no. 2, pp. 494–504, 2011.

[9] S. Friedland, Q. Li, and D. Schonfeld, "Compressive sensing of sparse tensors," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4438–4447, 2014.

[10] J. D. Trzasko and A. Manduca, "A unified tensor regression framework for calibrationless dynamic, multi-channel mri reconstruction," in *Proc. Int. Soc. Magn. Reson. Med*, 2013, p. 603.

[11] Y. Yu, J. Jin, F. Liu, and S. Crozier, "Multidimensional compressed sensing mri using tensor decomposition-based sparsifying transform," *PloS one*, vol. 9, no. 6, p. e98441, 2014.

[12] J. D. Trzasko, "Exploiting local low-rank structure in higher-dimensional mri applications," in *Wavelets and Sparsity XV*, vol. 8858. SPIE, 2013, pp. 551–558.

[13] J. He, Q. Liu, A. G. Christodoulou, C. Ma, F. Lam, and Z.-P. Liang, "Accelerated high-dimensional mr imaging with sparse sampling using low-rank tensors," *IEEE transactions on medical imaging*, vol. 35, no. 9, pp. 2119–2129, 2016.

[14] S. F. Roohi, D. Zonoobi, A. A. Kassim, and J. L. Jaremko, "Multi-dimensional low rank plus sparse decomposition for reconstruction of under-sampled dynamic mri," *Pattern Recognition*, vol. 63, pp. 667–679, 2017.

[15] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009. [Online]. Available: https://doi.org/10.1137/07070111X