DMD WITH LOW RANK FEATURE COMPRESSION FOR VIDEO BACKGROUND EXTRACTION [*]

HAOZE HE

1. Introduction.

DMD, a mathematical method that was developed to understand the evolution of a dynamical system without prior knowledge of their governing equations, proves to be well-suited for extracting insights from real-world data captured as snapshots over time. Video data falls into this scheme, and therefure DMD is a suitable method to understand video data. Additionally, another property of DMD, wwhich we will delve into later, is the higher the dimension of the snapshot/observation/measurement is, the better DMD can understand or reconstruct the underlying dynamics. Video data typically falls into this category, with each frame representing an image residing in a high-dimensional Euclidean space. The specific focus of this report is on background extraction in video learning tasks.

2. DMD and video background extraction.

2.1. DMD.

Consider a discrete dynamical system

$$z_i = T(z_{i-1})$$

where the states $z_i, i = 0, 1, \ldots$ lie in a state space $\mathcal{X}$. Consider observables (e.g., measurements) $f \in \mathcal{F} : \mathcal{X} \to \mathbb{C}$ and we define the Koopman operator, which is a key concept of DMD, as follows:

$$\mathcal{K} f(z_i) = f(T(z)) = f(z_{i+1}).$$

By its definition, $\mathcal{K}$ is linear, and if the space of observalbes $\mathcal{F}$ is "nice" enough, (e.g., $\mathcal{F} \subseteq L^2(X, \mu)$), $\mathcal{K}$ has a discrete spectral decomposition, and DMD is a way to approach this spectral decomposition given only snapshot data of the dynamics.

Let $S = [s_1 \cdots s_n] \in \mathbb{C}^{d\times n}$ be the snapshots, that is, $s_i = [f_1(z_i) \ \cdots \ f_d(z_i)]$; they are the measurements of the states $z_1, \ldots, z_n$. Let $X = S(1 : n-1), Y = S(2 : n)$, our task is to find a best fit matrix A such that $\|X^T A - Y^T\|_F$ is as small as possible. Mathematically, one of the optimal A can be computed by $A = (YX^\dagger)^T$ where $\dagger$ denotes the pseudo-inverse. However, computing the pseudo–inverse of a rank–deficient matrix is unstable. To avoid the ill–conditioness, we apply the Schmid DMD. Let $U_k \Sigma_k V_k^{\mathsf{H}}$ be the truncated rank–k SVD of X, the Schmid DMD computes an approximation of A^T in span$\{U_k\}$ as

$$A^T = U_k U_k^{\mathsf{H}} Y (U_k \Sigma_k V_k^{\mathsf{H}})^\dagger U_k = U_k^{\mathsf{H}} Y V_k \Sigma_k^{-1}.$$

We also provide our own DMD implementation with normalization.

The above computational approach in finite dimension has its counter part in functional space. It is equivalent to first restrict $\mathcal{K}$ to the subspace spanned by $f_1, \ldots, f_d$ where we assume

$$s_i = [f_1(t_i) \cdots f_d(t_i)]^T,$$

with $f_1, \ldots, f_d$ are not fully known but only known their evaluations at $t_1, \ldots, t_n$ (we can think of t_i as time). After the restriction, the above DMD approach corresponds

[*]Joint work with Nian Shao.

to the following least square problem

$$\int |fA - \mathcal{K}f|^2 d\delta_N$$

where δ_N is the discrete measure, $f = [f_1 \cdots f_d]$ is a quasi-matrix in the sense of [5]. From this discussion, we can expect that our computed A's eigenvalues approximate the true eigenvalues of $\mathcal{K}$ better and better with more snapshots (i.e., more time steps so that the discrete measure approximates the underlying measure of the sample space) and more observables (i.e., larger subspaces so that the restricted map approximates the original map). Then the eigenvalue of A is the approximated eigenvalues of $\mathcal{K}$ with Rayleigh-ritz procedure to restrict to span$\{f_1, \cdots, f_d\}$.

Let $A = Q\Lambda Q^{-1}$ be the corresponding eigenvalue decomposition of A. From the minimization process, we know that

$$\mathcal{K}f \approx fA,$$

thus,

$$\mathcal{K}fQ \approx fQ\Lambda,$$

and we know that $\phi = [\phi_1 \cdots \phi_d] = fQ$ are the approximated eigenfunctions of $\mathcal{K}$. But they are only known to us at the given snapshots, i.e., we only know $\phi(t_1), \dots, \phi(t_n)$.

Then we can express the partially known observables f in terms of ϕ,

$$f^T = Q^{-T}(fQ)^T = Q^{-T}\phi^T = \sum z_i\phi_i$$

where z_i is the ith columns of Q^{-T}. Then it follows straightforwardly to obtain the so-called koopman mode decomposition

$$\mathcal{K}^k f^T(s_1) \approx \sum z_i\phi_i(s_1)\lambda_i^k.$$

The vector $z_i\phi_i(s_1)$ has key information of our dynamical system, especially for the task of video background extraction, which will be discussed in the following subsection.

2.2. Video background extraction. Consider the vector z_i, each entry of z_i, $z_i(j)$, is the component that how much of f_j belongs to the eigenfunction ϕ_i, if we collect all of these component into one vector $z_i \in \mathbb{C}^d$, it extracts the component that how our snapshot $f = [f_1 \cdots f_d]$ belongs to the ith eigenfunction ϕ_i. Note that this vector z_i will envolve with respect to time in the same way as the eigenfunction ϕ_i envolve with respect to the koopman operator $\mathcal{K}$ by the formula

$$\mathcal{K}^k f^T(s_1) \approx \sum z_i\phi_i(s_1)\lambda_i^k.$$

That is, we decompose the whole dynamical system in terms of the individual envolution of the components corresponding to each eigenfuction ϕ_i.

Consider our data now as video and each snapshot is an image, the z_i vector is also an image. It captures how much each frame can be decomposed into different eigenfuntcion ϕ_i. But we can have physical interpretation for z_i in terms of video. Imagine one extreme case that ϕ_1 has eigenvalue 1, then the component that corresponding to ϕ_1, z_1, will not change with respect to time, i.e., z_i capture the unmoved component in the video, which by definition, is the background. For the component vector z_i that corresponds to an eigenvalue of $\mathcal{K}$ with a large negative real part, it will decay rapidly and therefore corresponding to the object that disappear quickly in the video. For example, when a car go pass by a section of the highway, we can expect that the components that corresponding to the car, if we express them in koopman modes, shall correspond to eigenvalues with large negative real parts. On the other hand, the components that corresponding to the road should close to 1. These key observation provide us with a criteria to select koopman modes as background or foreground; we can look at the quantity $|\log(\lambda_i)|$, if this quantity is small, meaning that the corresponding koopman mode has approximated eigenvalue close to 1, which is more likely to correspond to the background.

3. Our approach.

3.1. Low rank image patches. For video data, one major challenge is that each frame is a image with a very high dimension, e.g., 1920×1080. To efficiently perform DMD even in real time, we should compress our video data. In [1–3], randomized SVD (RSVD) are proposed to replace the original SVD in the DMD computation to reduce the complexity. However, they all consider the standard vectorization of images, which might destroy the connection between nearby pixels within one frame . This problem is addressed in [4] by dictionary learning, where images are divided into small patches, and for each small patch, a dictionary is learnt from the whole video, and in each frame, we only extract the coefficients of the corresponding components in the dictionary, this reduces the dimension and keep the local coherence of the pixels. However, in [4], the process of the dictionary learning is solved by very expensive iterative optimization method, which is much more expensive than the full DMD directly to frames.

In this report, we propose Low Rank Image Patch. It is based on the observation that if we vectorize each patch, stack each vectorized patch at different frames in the video into a short and fat matrix, this matrix is expected to be low rank, as for each patch, it is very unlikely to change dramatically with respect to time, (e.g., consider a car passing by a section of a road), and therefore we can use only a few basis vector to approach the range. This procedure is summarized as follows

Algorithm 3.1 Low Rank Patch Extraction (LRPE)

```
Input: A video $V \in \mathbb{R}^{m\times n\times T}$, # of patches $N$, rank $r$,
Output: $r$ basis vectors of each patch, in total $r \times N$ vectors.
1: for i = 1 to N
2:     extract patch $i$, $p_i(k)$ for $V(:,:,k), k = 1, \dots, T$,
3:     $A = [p_i(1), \dots, p_i(T)]$
4:     compute top $r$ left singular vectors of $A$ and keep them for output
5: end for
```

After finding the basis vectors, we only need to run DMD on the coefficient of each patch with respect to the corresponding basis vectors. That is, we perform usual DMD on the vectorization of the coefficients of each patch.

Algorithm 3.2 LRPE DMD

```
Input: A video $V \in \mathbb{R}^{m\times n\times T}$, basis vectors of $N$ patches $Q_1,\dots,Q_N$
1: for i = 1 to N
2:     extract patch $i$, $p_i(k)$ for $V(:,:,k), k=1,\dots,T$,$C_i(k)=Q_i^T p_i(k)$
3: end for
4: Run DMD on $[[C_1(1)\cdots C_N(1)]^T\cdots[C_1(T)\cdots C_N(T)]^T]$.
```

3.2. Iterative refinement. It is possible to refine the result by iteratively running DMD on the residual after extracting the background, to capture more part of moving background in the resulting foreground, as suggested in [].

However, we find out that if we do not do truncation in solving the least square problem in DMD, we should expect identical result among different refinement iteration. If we have

$$A = YX^{\dagger}$$

and obtain the first background as

$$B = [v_1 \ \cdots \ v_n].$$

Divide B into

$$B_1 = [v_1 \ \cdots \ v_{n-1}], \quad B_2 = [v_2 \ \cdots \ v_n] = AB_1$$

and substract them from the original snapshots we get

$$X' = X - B_1, \quad Y' = Y - AB_1.$$

We will have

$$Y'X'^{\dagger} = A.$$

However, if we truncate the SVD when we solve the least square problem in DMD, which corresponds to throwing away information we do not trust, we can expect improvement from the.

4. Numerical results. All numerical experiments in this section are implemented in Matlab 2022b and executed with an AMD Ryzen 9 6900HX Processor (8 cores, 3.3–4.9 GHz) and 32 GB of RAM. We apply our novel DMD background extraction method to two real videos, one is of small size (90×72 resolution) and the other is of large size (852×480 resolution). We also compared our implementation with the original implementation of dictionary learning [] on the small size video, while on the large-size video, the original algorithm is too slow to produce an output within a reasonable time.

For the small-size video, we use their video as Figure 4.1. Compared with their algorithm (using their public codes), we can achieve a $484\times$ speed–up, i.e., from 141.7 seconds to 0.2927 seconds. The background and foreground reconstructed are given in Figures 4.2 and 4.3. Compared with their results, our method can recover a better foreground, i.e., two people are walking around.

Now we would like to discuss possible reasons for the superiority of low–rank approximation (our method) than dictionary learning (their method). Dictionary learning and sparse approximation are effective methods in signal processing. However, compared with signal, the video is much more complex and noiseless. In our setting, for each batch, there is no need to use a very large size of dictionary. Moreover, in our problem, we do not to predict the further video, which means we have all data in hand. In such a situation, truncated SVD is the best low–rank approximation, especially when the video admits a low–rank structure. Thus, it is not surprise that low–rank approximation will outperform dictionary learning.

For the large-size video (Figure 4.4), their method does not work; it does not produce the result within a reasonable time. We sucessfully extract the backgound keyboard of this instrument, as shown in Figure 4.5. We only show one image because this does not change with respect to frames in this example.

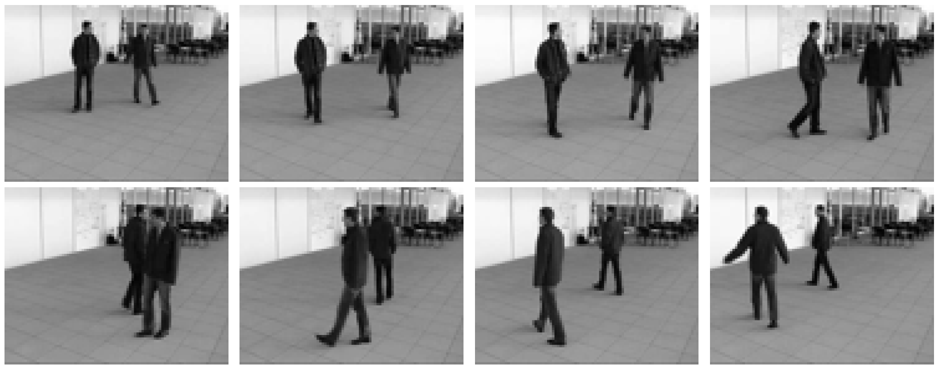
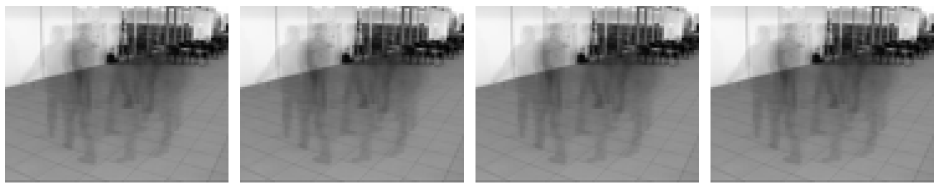
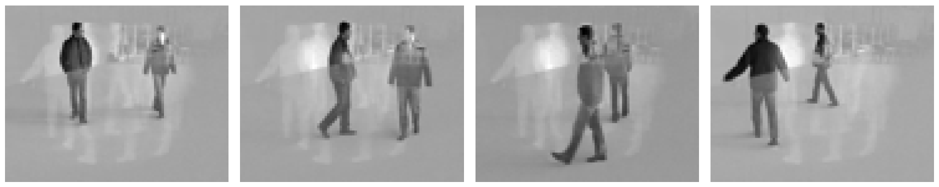


Fig. 4.1: Slice of small size video.



(a) Background

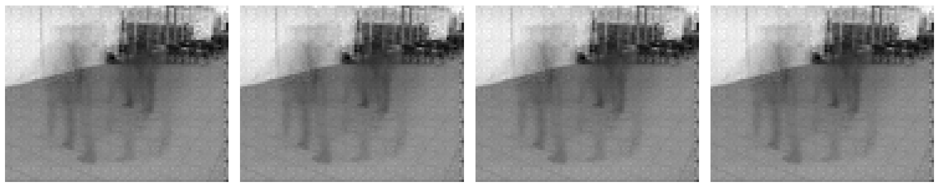


(b) Foreground

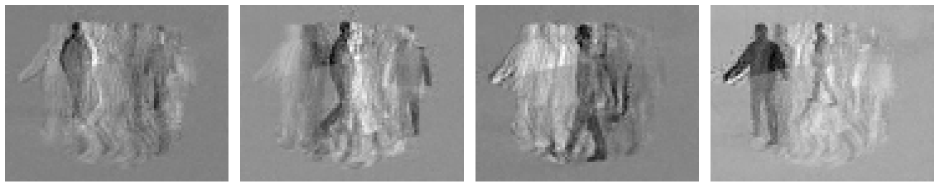
Fig. 4.2: Background and foreground recovered by our method.

5. Conclusion. In this report we present a novel DMD algorithm for video background extraction, and we demonstrate the intuition why low rank patch compression can help improve both the accuracy and efficiency. We demonstrate our advantage over the original improvement on real videos.

During the low rank patch compression phase, the current compression method is standard SVD, but in this situation, the target rank is known to us, and therefore



(a) Background



(b) Foreground

Fig. 4.3: Background and foreground recovered by their method.

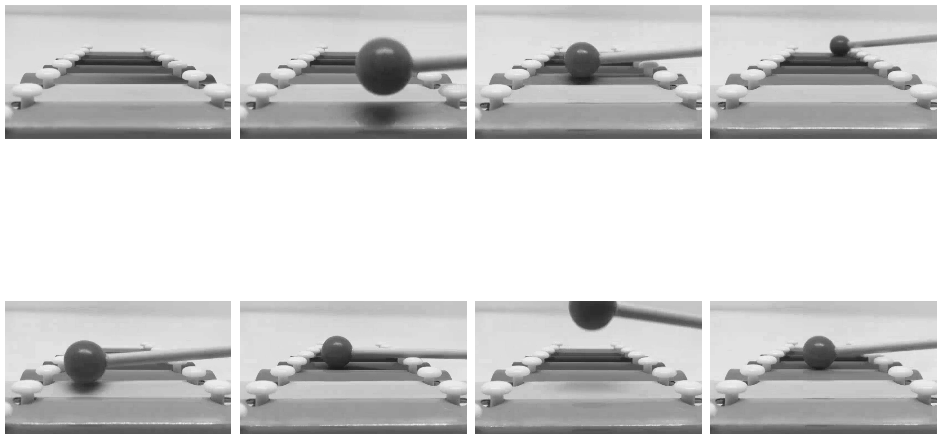


Fig. 4.4: Slice of small size video.

we expect Randomized SVD (RSVD) can significantly improve the efficiency without too much loss of accuracy.

We also would like to extend our current algorithm into streaming setting to capture the change of the background with respect to time, and this we need more efficient implementation of compression (again, RSVD is suitable) and seek for SVD or QR update to allow for streaming data.

We also expect that our work can be extended to video frames prediction.

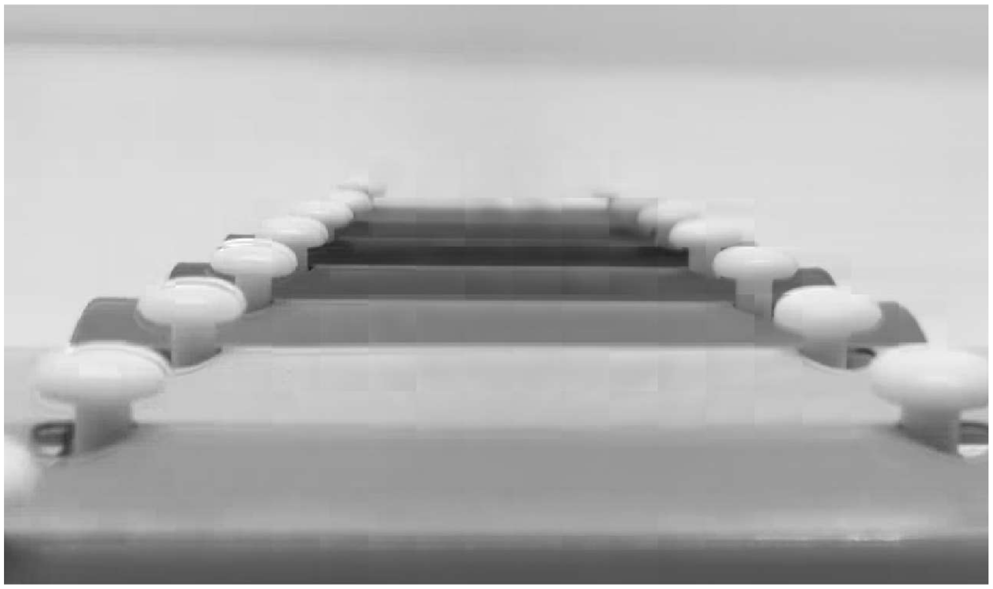


Fig. 4.5: Background of the large video

REFERENCES

[1] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16(5):1479–1492, 2019.

[2] N. B. Erichson and C. Donovan. Randomized low-rank dynamic mode decomposition for motion detection. *Computer Vision and Image Understanding*, 146:40–50, 2016.

[3] J. Grosek and J. N. Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*, 2014.
[4] I. U. Haq, K. Fujii, and Y. Kawahara. Dynamic mode decomposition via dictionary learning for foreground modeling in videos. *Computer Vision and Image Understanding*, 199:103022, 2020.
[5] L. N. Trefethen. Householder triangularization of a quasimatrix. *IMA J. Numer. Anal.*, 30(4):887–897, 2010.