Incremental Fixed-Rank Robust PCA for Video Background Recovery

Jian Lai, Wee Kheng Leow, and Terence Sim

Dept. of Computer Science, National University of Singapore
laij, leowwk, tsim@comp.nus.edu.sg

Abstract. Video background recovery is a very important task in computer vision applications. Recent research offers robust principal component analysis (RPCA) as a promising approach for solving video background recovery. RPCA works by decomposing a data matrix into a low-rank matrix and a sparse matrix. Our previous work shows that when the desired rank of the low-rank matrix is known, fixing the rank in the algorithm called FrALM (fixed-rank ALM) yields more robust and accurate results than existing RPCA algorithms. However, application of RPCA to video background recovery requires that each frame in the video is encoded as a column in the data matrix. This is impractical in real applications because the videos can be easily larger than the amount of memory in a computer. This paper presents an algorithm called iFrALM (incremental fixed-rank ALM) that computes fixed-rank RPCA incrementally by splitting the video frames into an initial batch and an incremental batch. Comprehensive tests show that iFrALM uses less memory and time compared to FrALM. Moreover, the initial batch size and batch quality can be carefully selected to ensure that iFrALM reduces memory and time complexity without sacrificing accuracy.

Keywords: Background recovery, incremental SVD, fixed-rank robust PCA.

1 Introduction

Video background recovery is a very important task in applications such as video surveillance, traffic monitoring, etc. Traditionally, various approaches have been developed for this task. Background recovery is closely related to moving object detection and some works [10, 12–14] attempt to simultaneously solve these two problems within one framework. Recent research offers robust principal component analysis (RPCA) as a promising alternative approach for solving a wide range of problems including video background recovery [2, 16]. RPCA utilizes the fact that the image frames in a video contain consistent information about the common background. It constructs a *data matrix* from multiple video frames and decomposes it into a *low-rank matrix* and a *sparse matrix*, such that the low-rank matrix corresponds to the background in the images and the sparse matrix captures the non-background components. It has been proved that exact

solution of RPCA is available if the data matrix is composed of a sufficiently low-rank matrix and a sufficiently sparse matrix [2, 3, 8, 15, 19]. Various algorithms have been proposed for solving RPCA problems [6, 8, 9, 15, 16]. In [6], we show that when the desired rank of the low-rank matrix is known, fixing the rank in the algorithm yields more robust and accurate results than the method based on *augmented Lagrange multiplier* (ALM), which is among the most efficient and accurate methods [8]. In particular, our fixed-rank algorithm called FrALM is less sensitive than ALM to the choice of the weighting parameter λ.

Application of RPCA to video background recovery requires that each frame in the video is encoded as a column in the data matrix. This is impractical in real applications because the videos can be easily larger than the amount of memory in a computer. For online application where the video frames are received continuously over long duration, this limitation is especially severe.

This paper presents an algorithm called iFrALM (incremental fixed-rank ALM) that computes fixed-rank RPCA incrementally by splitting the video frames into an *initial batch* and an *incremental batch*. Instead of reserving memory for all the video frames, iFrALM requires only enough memory to keep the incremental batch of video frames and the fixed-rank components of the initial batch. As new video frames arrive, iFrALM accumulates them into a fixed-size batch and uses them to update the results, thus overcoming memory limitation. Moreover, with prudent choices of batch size and content, iFrALM can execute more efficiently than our previous FrALM without sacrificing accuracy.

2 Related Work

2.1 Incremental PCA and RPCA

Singular value decomposition (SVD) is a powerful tool that is used in PCA and RPCA. Its computational cost on a $m \times n$ ($m > n$) matrix is $O(mn^2)$, which limits its application to small data set. To overcome this limitation, incremental SVD (iSVD) has been studied and many methods [1, 5, 17] are proposed. With iSVD, both computation time and memory are greatly saved.

One successful application of iSVD is incremental PCA, which gives rise to the eigenspace update algorithm [4] and Sequential Karhunen-Loeve method [7]. However, approximation errors of these methods cannot be estimated. To tackle this problem, Zhao et al. [18] proposed a SVD updating based approach for incremental PCA, which has a mathematically proven error bound.

Incremental SVD has also been used for RPCA. Rodriguez et al. [11] used iSVD in their incremental principal component pursuit (iPCP) algorithm but implementation details are not discussed. As iPCP updates the low-rank and sparse components by appending only a single column to the existed data, its accuracy cannot be guaranteed. To our best knowledge, no incremental method for fixed-rank RPCA has been proposed. In this paper, we show that proper application of iSVD can increase efficiency without sacrificing accuracy.

3 Incremental Fixed-Rank RPCA

Our original fixed-rank RPCA algorithm FrALM [6] solves the problem

$$\min_{\mathbf{A},\mathbf{E}} \|\mathbf{E}\|_F, \text{ subject to } \text{rank}(\mathbf{A}) = \text{known } r, \ \mathbf{D} = \mathbf{A} + \mathbf{E}, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm, and $\mathbf{D}, \mathbf{A}$ and $\mathbf{E} \in \Re^{m\times n}$ are data matrix, rank-r matrix and noise matrix, respectively. It uses SVD to minimize Eq. 1. SVD factorizes a matrix $\mathbf{A} \in \Re^{m\times n}$ as $\mathbf{USV}^\top$, where $\mathbf{S} \in \Re^{n\times n}$ is a diagonal matrix whose diagonal elements are singular values, and $\mathbf{U} \in \Re^{m\times n}$ and $\mathbf{V} \in \Re^{n\times n}$ are the left and right singular matrix, respectively. In applications such as video background recovery, the number of pixels m is much greater than the number of frames n, and essential information lies in a significantly low-dimensional space defined by the first r (i.e., $r \ll n$) dominant singular vectors and values. Thus, $\mathbf{A}$ can be well approximated by the rank-r SVD

$$\mathbf{A}^r = \mathbf{U}^r\mathbf{S}^r{\mathbf{V}^r}^\top, \tag{2}$$

where the diagonal elements of $\mathbf{S}^r \in \Re^{r\times r}$ are the largest r singular values, and $\mathbf{U}^r \in \Re^{m\times r}$ and $\mathbf{V}^r \in \Re^{n\times r}$ are the matrices consisting of the corresponding r left and right singular vectors, respectively.

Given an incremental batch of data encoded in the matrix $\mathbf{D}_i \in \Re^{m\times l}$, the rank-$r$ SVD of the combined matrix $[\mathbf{A}^r\ \mathbf{D}_i]$ can be computed using incremental SVD (iSVD) [17], which works on $\mathbf{U}^r$, $\mathbf{S}^r$, and $\mathbf{V}^r$ instead of $\mathbf{A}^r$. Consequently, the computation cost of iSVD is $O(ml^2)$, which is much smaller than that of applying normal SVD on the combined matrix, which is $O(m(n+l)^2)$.

Our incremental fixed-rank RPCA algorithm works on the data matrix that is split into an initial batch $\mathbf{D}_0$ and an incremental batch $\mathbf{D}_1$. It applies the non-incremental FrALM on the initial batch $\mathbf{D}_0$ to recover rank-r matrices $\mathbf{U}_0$, $\mathbf{S}_0$, and $\mathbf{V}_0$. Then, it applies incremental fixed-rank ALM algorithm (iFrALM, Algorithm 1) to compute the rank-r solutions of the combined matrix $[\mathbf{A}_0\ \mathbf{D}_1]$, where $\mathbf{A}_0 = \mathbf{U}_0\mathbf{S}_0\mathbf{V}_0^\top$, and produces rank-$r$ matrices $\mathbf{U}_1$, $\mathbf{S}_1$, and $\mathbf{V}_1$. Consequently, the rank-r component of $[\mathbf{A}_0\ \mathbf{D}_1]$ can be recovered as $\mathbf{A}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^\top$. This incremental process can be repeated for additional incremental batches $\mathbf{D}_i, i > 1$. In general, iFrALM takes $\mathbf{U}_{i-1}$, $\mathbf{S}_{i-1}$, $\mathbf{V}_{i-1}$ and $\mathbf{D}_i$ as inputs and produces $\mathbf{U}_i$, $\mathbf{S}_i$ and $\mathbf{V}_i$ as outputs, which can be used in the next incremental step.

The iFrALM algorithm has a similar structure as the non-incremental FrALM. In line 2, $\text{sgn}(\cdot)$ is the sign function which computes the sign of each matrix element, and $J(\cdot)$ computes a scaling factor

$$J(\mathbf{X}) = \max(\|\mathbf{X}\|_2,\ \lambda^{-1}\|\mathbf{X}\|_\infty) \tag{3}$$

as recommended in [8]. T_ϵ in line 7 denotes the soft-thresholding function

$$T_\epsilon(x) = \begin{cases} x - \epsilon, \text{ if } x > \epsilon, \\ x + \epsilon, \text{ if } x < -\epsilon, \\ 0, \quad \text{otherwise.} \end{cases} \tag{4}$$

Algorithm 1: iFrALM

Input: $\mathbf{U}_{i-1}$, $\mathbf{S}_{i-1}$, $\mathbf{V}_{i-1}$, $\mathbf{D}_i$, r, and λ.

1 $\mathbf{A} = 0$, $\mathbf{E} = 0$.
2 $\mathbf{Y} = sgn(\mathbf{D}_i)/J(sgn(\mathbf{D}_i)), \mu > 0, \rho > 1$.
3 **repeat**
4 **repeat**
5 $\mathbf{U}_i, \mathbf{S}_i, \mathbf{V}_i^\top = \text{iSVD}_r([\mathbf{A}_{i-1} \;\; \mathbf{D}_i - \mathbf{E} + \mathbf{Y}/\mu])$.
6 $\mathbf{A} = \mathbf{U}_i\, \mathbf{S}_i\, \mathbf{V}_i(n+1 : n+l, :)^\top$.
7 $\mathbf{E} = T_{\lambda/\mu}(\mathbf{D}_i - \mathbf{A} + \mathbf{Y}/\mu)$.
8 **until** *convergence*;
9 $\mathbf{U}', \mathbf{S}', \mathbf{V}' = \text{SVD}(\mathbf{S}_i\, \mathbf{V}_i(1 : n, :)^\top)$.
10 $\mathbf{U}_{i-1} = \mathbf{U}_i\mathbf{U}'$, $\mathbf{S}_{i-1} = \mathbf{S}', \mathbf{V}_{i-1} = \mathbf{V}'$.
11 $\mathbf{Y} = \mathbf{Y} + \mu(\mathbf{D}_i - \mathbf{A} - \mathbf{E}), \mu = \rho\mu$.
12 **until** *convergence*;

Output: $\mathbf{U}_i, \mathbf{S}_i, \mathbf{V}_i$.

In line 5, a rank-r iSVD (iSVD$_r$) [17] is used to compute the SVD of the combined matrix $[\mathbf{A}_{i-1} \;\; \mathbf{D}_i - \mathbf{E} + \mathbf{Y}/\mu]$. It computes the rank-$r$ matrices $\mathbf{U}_i$, $\mathbf{S}_i$, and $\mathbf{V}_i$ from $\mathbf{U}_{i-1}$, $\mathbf{S}_{i-1}$, and $\mathbf{V}_{i-1}$ instead of directly from $\mathbf{A}_{i-1} = \mathbf{U}_{i-1}\mathbf{S}_{i-1}\mathbf{V}_{i-1}^\top$. Next, lines 6 and 7 compute the low-rank matrix $\mathbf{A}$ and error matrix $\mathbf{E}$ of $\mathbf{D}_i$, which are represented by the last l rows of $\mathbf{V}_i$. It is not necessary to compute the error matrix of $\mathbf{A}_{i-1}$ because $\mathbf{A}_{i-1}$ is already a low-rank matrix.

Lines 9 and 10 are used to update $\mathbf{U}_{i-1}$, $\mathbf{S}_{i-1}$, and $\mathbf{V}_{i-1}$. Without them, the subspaces spanned by $\mathbf{U}_i$ will be identical to that of $\mathbf{U}_{i-1}$, which defeats the incremental algorithm. These matrices can be updated either in the inner loop or the outer loop. Updating in the inner loop before it converges may cause instability and incur additional computation cost. Therefore, we choose to update them in the outer loop. Note that $\mathbf{U}_i\, \mathbf{S}_i\, \mathbf{V}_i(1 : n, :)^\top$ is an $m \times n$ matrix, which is much larger than the $r \times n$ matrix $\mathbf{S}_i\, \mathbf{V}_i(1 : n, :)^\top$. Therefore, line 9 performs SVD on the smaller matrix $\mathbf{S}_i\, \mathbf{V}_i(1 : n, :)^\top$, whose results are used to update $\mathbf{U}_{i-1}$, $\mathbf{S}_{i-1}$, and $\mathbf{V}_{i-1}$ as shown in line 10. Finally, line 11 is the standard technique for applying augmented Lagrange multiplier (ALM) method.

The time complexity of iFrALM is dominated by the incremental SVD, which is $O(ml^2)$. Its memory complexity is $O(ml)$, the amount required for storing $\mathbf{D}_i$, $\mathbf{A}$, $\mathbf{E}$, and $\mathbf{Y}$. Thus, it uses less time and memory compared to FrALM on the whole combined matrix $[\mathbf{A}_{i-1} \; \mathbf{D}_i]$, which has time and memory complexity of $O(m(n+l)^2)$ and $O(m(n+l))$, respectively.

4 Experiments and Discussions

4.1 Data Preparation

The performance of the proposed iFrALM was tested on recovering the stationary backgrounds of three videos: Kungfu [6], Traffic [6] and Shopping Center (available from http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html). These

color videos were converted to gray images. For Shopping Center video, the gray images were downsampled to 128×160. The other videos had image size of 150×200. Test programs were implemented in Matlab and ran in a 64-bit Windows 7 PC with Intel Core i7-2600 CPU and 16GB RAM.

Four test scenarios were set by varying the following properties:

1. initial batch size (Section 4.2),
2. incremental batch size (Section 4.3),
3. batch quality (Section 4.4), and
4. multiple incremental batches (Section 4.5).

Both iFrALM and FrALM were executed under these four conditions. In addition, the average image of each video was computed to indicate the complexity of a video. Compared to the ground truth, average image with small error corresponds to a simple problem.

Once the background is recovered, the moving objects can be easily detected by thresholding the differences between the input image and the recovered background $\mathbf{A}$. Therefore, we also compare iFrALM with some baselines of moving object detection in Section 4.6.

As all videos were captured with stationary cameras, the fixed rank r was 1 for both FrALM and iFrALM. The parameters λ, ρ and initial μ were fixed to $1/\sqrt{m}$, 1.5 and $0.5/\|\mathbf{Y}\|_2$. The algorithms' accuracy was measured in terms of the mean squared error (MSE) between the ground truth $\mathbf{G}$ and the algorithm outputs $\mathbf{U}_i$, $\mathbf{S}_i$, and $\mathbf{V}_i$:

$$E_g = \frac{1}{mn}\|\mathbf{G} - \mathbf{U}_i\mathbf{S}_i{\mathbf{V}_i}^\top\|_F^2. \tag{5}$$

For Kungfu video, the background only image was available and it served as the ground truth. For Shopping Center and Traffic videos, their ground truths were not available. In this case, FrALM was applied to the whole video and the recovered rank-1 matrix was regarded as the ground truth. In addition, execution time of FrALM and iFrALM were recorded.

4.2 Initial Batch Size

In this experiment, 70 frames each of Shopping Center and Kungfu videos were used. Their initial batch size varied from 5 to 50 consecutive frames in increments of 5, and their incremental batch size was fixed at 20 consecutive frames. The Traffic video had 250 frames. So, its initial batch size was set to 10, 20, 50, 100, 150, and 200, and its incremental batch size was fixed at 50 frames.

Figure 1(a) shows the results of running FrALM on all video frames in the initial and incremental batches, FrALM on the initial batch, iFrALM on the incremental batch, and average of all video frames. Given enough video frames, the fixed-rank methods are much more accurate than the average frame. Moreover, the error (MSE) curves of iFrALM and FrALM on initial batch have the same trend. As the initial batch size increases, iFrALM's error decreases and approaches that of FrALM on all video frames.

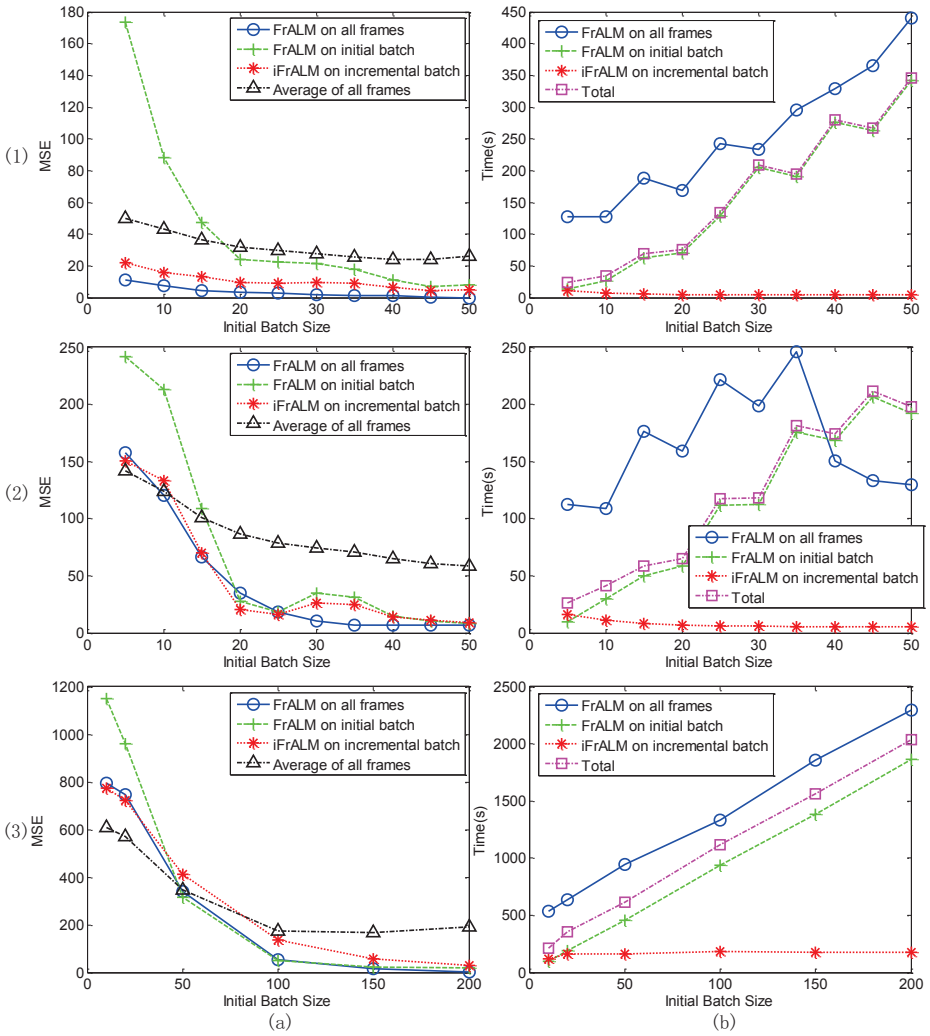

Fig. 1. The results of varying initial batch size. (1) Shopping Center. (2) Kungfu. (3) Traffic. (a) MSE. (b) Running time.

Figure 1(b) shows the algorithms' running time of the algorithms. Note that the results of FrALM on initial batch serve as inputs of iFrALM. So, the total execution time is sum of the execution times of FrALM on initial batch and iFrALM on the incremental batch. FrALM's running time increases with increasing initial batch size. On the other, iFrALM's running time is not significantly affected by the initial batch size. Moreover, it is much smaller than FrALM's running time on the initial batch. As a result, the total execution time is significantly smaller than FrALM's running time (in most cases) on all video frames.

4.3 Incremental Batch Size

This experiment used the same data as that in Section 4.2. For both Shopping Center and Kungfu videos, their initial batch size was fixed at 20 consecutive frames and their incremental batch size varied from 5 to 50 consecutive frames in increments of 5. For the Traffic video, its initial batch size was fixed at 50 and its incremental batch size was set to 10, 20, 50, 100, 150, and 200.

Figure 2 shows that the algorithms' performance is consistent with that in Section 4.2. In particular, iFrALM's error approaches that of FrALM on all frames as incremental batch size increases. The errors of both iFrALM and FrALM on all frames are small compared to that of average frame. FrALM's error on the initial batch is constant because there is no change in the initial batch size. iFrALM's running time is small compared to that of FrALM on the initial batch, and it increases more gradually with increasing incremental batch size compared to that of FrALM on all video frames. Therefore, the total execution time is smaller and increases more gradually than that of FrALM on all frames.

Figure 3 shows some sample results for the case where all incremental frames are processed. For the Shopping Center video, image averaging produces an acceptable result, indicating that this video is relatively easy to process. On the other hand, for the Kungfu and Traffic videos, image averaging produces visible "ghost" defects in the recovered background images, whereas FrALM and iFrALM produce cleaner background images, which is confirmed by the quantitative results shown in Fig. 2(a).

4.4 Batch Quality

The quality of the initial batch can affect the accuracy of iFrALM. Its quality is bad if it contains slowly moving or temporary stationary objects, which can be confused with the actual stationary background. On the other hand, its quality is good if it contains only stationary background or fast moving objects. For this test scenario, 50 frames of the Shopping Center videos and Kungfu videos were selected for both the initial and incremental batches. Four test cases were performed: (1) FrALM on a bad initial batch, whose results were fed to (2) iFrALM on a good incremental batch, (3) FrALM on a good initial batch, whose results were fed to (4) iFrALM on a bad incremental batch.

Table 1 shows that initial batch quality significantly affects result of iFrALM. With a bad initial batch, initial results fed to iFrALM is inaccurate. Although iFrALM can reduce error with good incremental batch, error is still large compared to the $4th$ case. On the other hand, with good initial results from FrALM, iFrALM working a bad incremental batch still produces a lower error compared to the $2nd$ case. This result is expected because iFrALM uses iSVD to obtain approximate solutions. So, good quality initial batch is crucial for iFrALM.

4.5 Multiple Incremental Batches

In online application, incoming video frames are accumulated into a batch and sent to the incremental algorithm. As time progresses, more video frames are

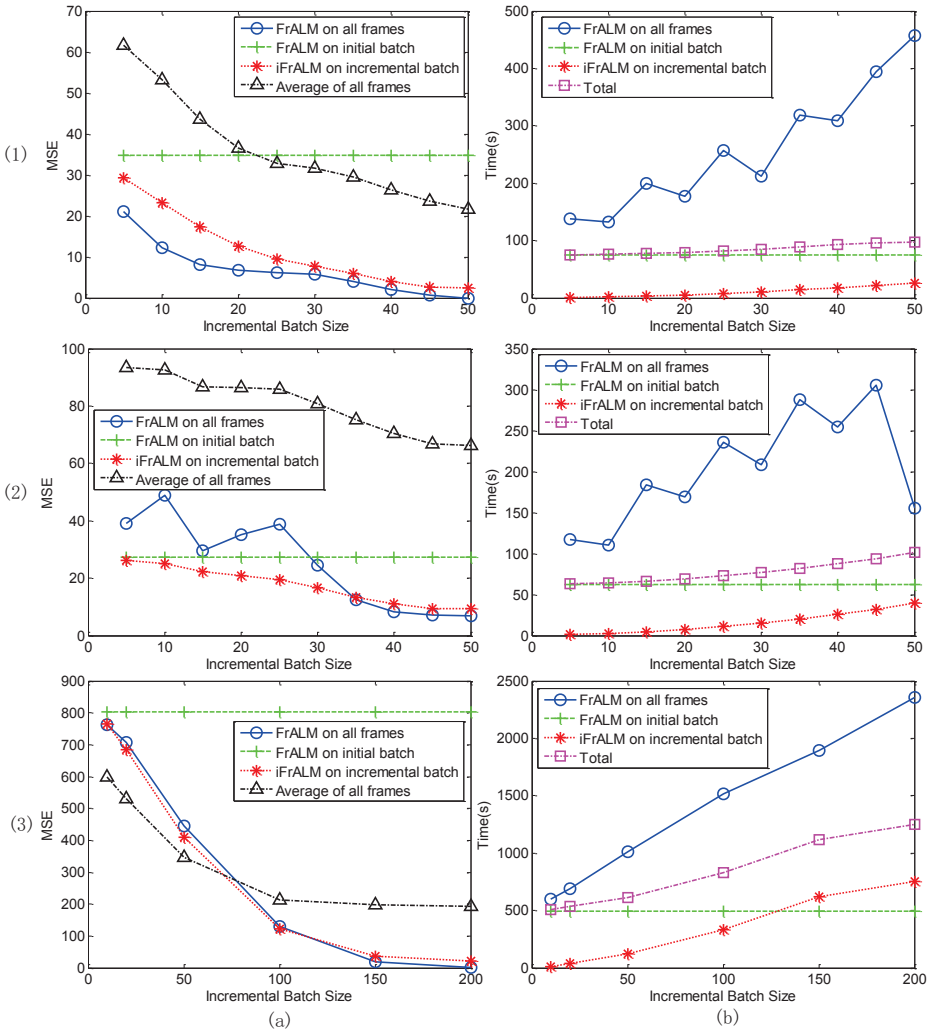

Fig. 2. The results of varying incremental batch size. (1) Shopping Center. (2) Kungfu. (3) Traffic. (a) MSE. (b) Running time.

accumulated and multiple batches are sent to the algorithm iteratively. This subsection describes a test performed under the same scenario using 1000 consecutive frames of the Shopping Center video. The first 50 frames formed the initial batch and was processed by FrALM to obtain the initial results for iFrALM. Subsequently, incremental batches of video frames were processed by iFrALM iteratively for the remaining 950 frames. The incremental batch size b varied for various test cases, namely 10, 20, 30, 40, 50, 100, and 200, and the number of iterations $k = \lceil 950/b \rceil$. Two incremental batch selection schemes were tested: (1)

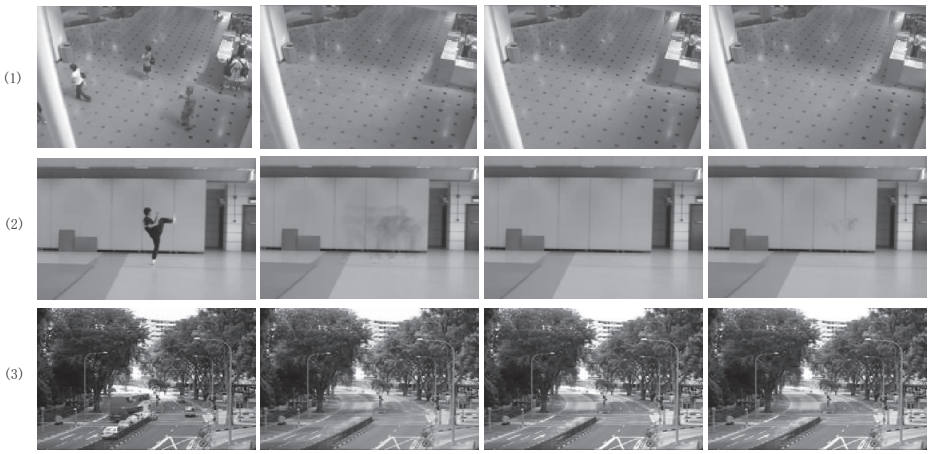

Fig. 3. Results of processing all image frames. (1) Shopping Center. (2) Kungfu. (3) Traffic. (a) Sample video frame. (b) Average frame. (c) FrALM. (d) iFrALM.

Table 1. Effect of batch quality of iFrALM's error (MSE).

Dataset	Shopping Center	Kungfu
FrALM with bad initial batch	101.8	139.3
iFrALM with bad initial and good incremental batch	35.1	48.5
FrALM with good initial batch	10.3	8.2
iFrALM with good initial and bad incremental batch	17.8	18.3

consecutive selection, (2) regular sampling at regular frame intervals. In comparison, regular sampling requires a large buffer for storing the incoming frames for selection at regular intervals, whereas consecutive selection requires a small buffer to store just the b consecutive frames.

Figure 4 plots MSE vs. execution time for each test case under the two selection schemes. Test result shows that smaller batch size b leads to lower total running time but higher MSE. This is because the computational cost of iFrALM is $O(km(l/k)^2) = O(ml^2/k) = O(mlb)$, which is proportional to b. With smaller sample size b, less information is available at each iteration, which leads to higher error.

Compared to consecutive selection, regular sampling captures more global information over the video. Therefore, it achieves a higher accuracy than consecutive selection at the same batch size b. Regular sampling takes more time

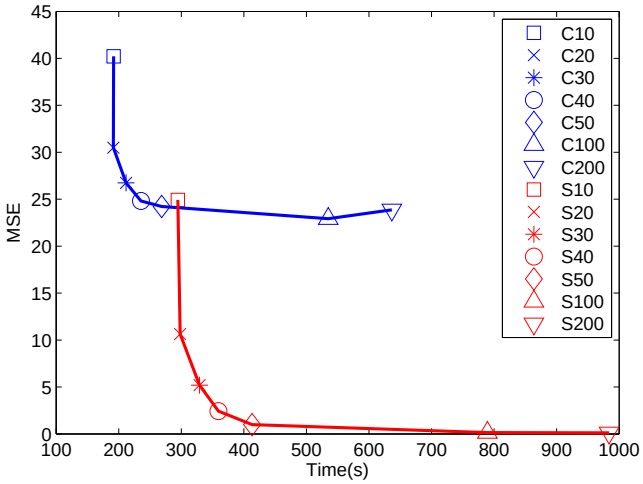

Fig. 4. Plot of MSE vs. execution time. 'C' denotes consecutive selection and 'S' denotes regular sampling. The number denotes incremental batch size b.

than consecutive selection because its iFrALM takes more iterations to converge. For all 1000 frames, FrALM takes 2640 seconds. Our iFrALM, though much faster than FrALM, is still not capable of handling background recovery problem in realtime, e.g., 30 frames per second at 1920×1080 pixels.

Figure 5 shows some recovered background images of this test. When the people, highlighted by the red rectangle in Fig. 5(1), remain at the same location for extended periods, consecutive selection scheme cannot remove them completely from the recovered background image (Fig. 5(2b–2d)). In contrast, regular sampling scheme can recover the stationary background well when b is at least 50 (Fig. 5(3c, 3d)).

4.6 Moving Object Detection

In this experiment, the first 100 frames of Kungfu video were used. The proposed iFrALM was compared with average, PCA [10], mixture of Gaussian (MoG) [12] and FrALM. For iFrALM, we split the 100 frames into 5 subsets by regular sampling. The first subset was used as initial batch and the other were used as multiple incremental batches. The results are shown in Figure 6. Thanks to multimodal modeling of background, MoG outperforms average and PCA. However, MoG cannot detect some interior pixels of the object. iFrALM and FrALM achieve similar results, which are much better than those of the others.

5 Conclusion

This paper presented an incremental method for computing fixed-rank robust PCA. The method works on the data matrix that is split into an initial batch

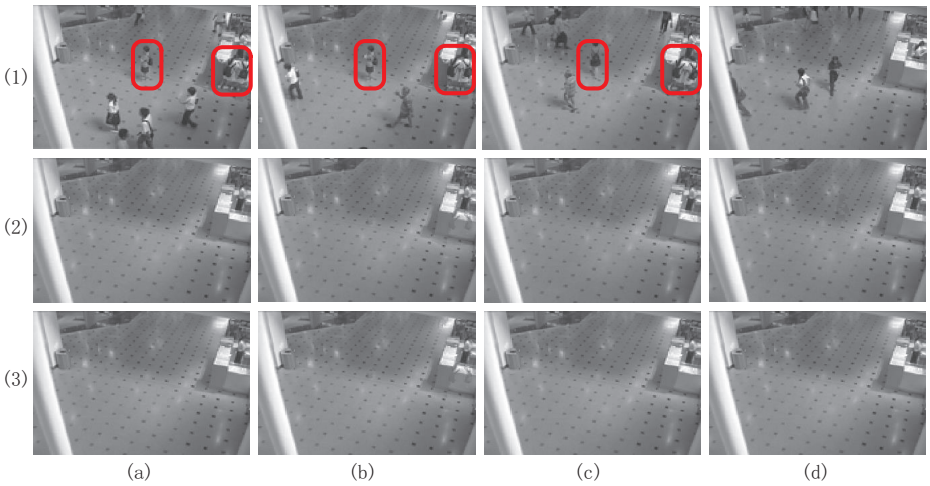

Fig. 5. Results processing multiple incremental batches. (1–3) show sample frames of Shopping Center video, iFrALM's results with consecutive selection and regular sampling, respectively. (a) Ground truth background. (b–d) show iFrALM's results with $b = 1, 50$ and 200, respectively.

Fig. 6. The detected moving objects of various methods on the 60th frame. The input frame is in Column 1. The results of average, PCA, MoG, FrALM and iFrALM are presented from Column 2 to 6, respectively.

and an incremental batch. It applies the non-incremental FrALM on the initial batch to recover a rank-r solution, which is used as the input for the incremental iFrALM to compute the rank-r solution of the incremental batch. iFrALM can be repeated for additional incremental batches iteratively. iFrALM uses incremental SVD to reduce computation time and SVD on a small matrix to reduce memory cost. Comprehensive tests were performed on three videos of varying lengths. Test results show that both iFrALM and FrALM produce more accurate results compared to video frame averaging, which is not robust. iFrALM uses less memory and time compared to FrALM. Test results also show that the initial batch size and batch quality can be carefully selected to ensure that iFrALM reduces memory and time complexity without sacrificing accuracy. These promising results thus pave the way for the application of incremental fixed-rank RPCA for online video background recovery.

Acknowledgement

This research is supported by MOE grant MOE2014-T2-1-062.

References

1. M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proc. ECCV*, 2002.
2. E. J. Candès, X. D. Li, Yi Ma, and J. Wright. Robust principal component analysis? *Journal of ACM*, 58(3):11, 2011.
3. E. J. Candès and Y. Plan. Matrix completion with noise. In *Proceedings of the IEEE*, pages 925–936, 2010.
4. S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321 – 332, 1997.
5. M. Gu and S. C. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Technical Report YALEU/DCS/RR-966, Yale Univ., 1994.
6. W. K. Leow, Y. Cheng, L. Zhang, T. Sim, and L. Foo. Background recovery by fixed-rank robust principal component analysis. In *Proc. CAIP*, 2013.
7. A. Levey and M. Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. *IEEE Trans. on Image Processing*, 9(8):1371–1374, 2000.
8. Z. C. Lin, M. M. Chen, L. Q. Wu, and Yi Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical Report UILU-ENG-09-2215, UIUC, 2009.
9. Z. C. Lin, A. Ganesh, J. Wright, L. Q. Wu, M. M. Chen, and Yi Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Proc. CAMSAP*, 2009.
10. N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on PAMI*, 22(8):831–843, 2000.
11. P. Rodriguez and B. Wohlberg. A matlab implementation of a fast incremental principal component pursuit algorithm for video background modeling. In *Proc. ICIP*, 2014.
12. C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. CVPR*, 1999.
13. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *Proc. ICCV*, 1999.
14. S. Varadarajan, L.J. Karam, and D. Florencio. Background recovery from video sequences using motion parameters. In *Proc. ICASSP*, 2009.
15. J. Wright, A. Ganesh, S. Rao, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Journal of ACM*, 2009.
16. J. Wright, Y. G. Peng, Y. Ma, A. Ganesh, and S. Rao. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In *Proc. NIPS*, 2009.
17. H. Y. Zha and H. D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.
18. H. T. Zhao, P. C. Yuen, and J.T. Kwok. A novel incremental principal component analysis and its application for face recognition. *IEEE Trans. on SMC, Part B: Cybernetics*, 36(4):873–886, 2006.
19. Z. H. Zhou, X. D. Li, J. Wright, E. J. Candès, and Yi Ma. Stable principal component pursuit. In *Proc. ISIT*, 2010.