MULTILINEAR ALGEBRA BASED TECHNIQUES FOR FOREGROUND AND BACKGROUND SEPARATION

BY NEHA TADIMETI

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Dr. Waheed U. Bajwa

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2017

ABSTRACT OF THE THESIS

Multilinear Algebra Based Techniques for Foreground and Background Separation

by Neha Tadimeti

Thesis Director: Dr. Waheed U. Bajwa

The work presented in this thesis aims to understand the use of tensor algebra for background and foreground separation in videos. Specifically, it tries to explore the advantages of tensor-based approaches over the vector-based ones. In vector-based approaches, video frames are vectorized and concatenated into columns of a matrix for foreground and background separation. Through vectorization, one cannot explore the multi-dimensional aspect of video frames. Recent research has shown that tensor algebra can be helpful in extracting useful information from a multi-dimensional perspective. In this thesis, we propose two new algorithms which use tensor algebra to solve for background and foreground separation.

In the first part of the thesis, we develop a mini-batch extension to Online Tensor Robust Principal Component Analysis (OTRPCA). The proposed extension significantly reduces the computational time in comparison to OTRPCA. It is also shown that the accuracy levels of background separation are higher than OTRPCA for a decent mini-batch size. As the mini-batch size further increases, accuracy levels fall as the dictionary update is one-shot and non-iterative.

In the second part of the thesis, online vector-based Grassmanian Robust Adaptive

Subspace Algorithm (GRASTA) is extended to tensor domain. The proposed Multi-Linear GRASTA (MLG) is also an online algorithm, thus suitable for real-time applications. Unlike the vector-based implementation, MLG explores the multi-dimensional nature of the video frames and solves for the separation problem across every dimension. MLG can process multiple frames at a time making it faster than other vector and tensor based separation algorithms. Detailed results are discussed which show that the accuracy of separation with MLG is competitive with the state-of-the-art.

Acknowledgements

I would like to express my most sincere gratitude to my adviser Dr. Waheed Bajwa who was extremely cooperative and guided me in this project. I sincerely thank him for giving me the freedom to choose my own path of research while steering me in the right direction when needed.

Many thanks to my thesis committee Dr. Elizabeth Torres, Dr. Zoran Gajic and Dr. Vishal Patel for offering their invaluable insights into my thesis.

I would also like to thank my fellow labmates of Information, Networks and Signal Processing Research (INSPIRE) lab for their support in my research. Special thanks to Tong Wu who had helped me in all the possible ways for my thesis.

Thank you to all my Professors and to Rutgers University for giving me the opportunity to pursue my Masters and providing me with state-of-the-art facilities.

Finally I would like to convey gratitude towards my parents, sister and my friends for their love and support. This would have been impossible without their support. Many thanks to my fiance Achyut Vemuri for always being on my side.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Subspace modeling has always been an important area of research [1–4]. Applications of subspace models are profound in computer vision, medical imaging, etc. With the wide spread ease of access to cameras, videos and images have become an important source of information. However, the efficiency of usage of this information is questionable. For instance, video surveillance cameras in most cases are visually monitored or are used to revisit an event after it has occurred. With the current existing technology, intelligent systems can be developed which automatically recognize activities and alert the user of any anomalies beforehand. With such and more wide range of applications in mind, a lot of research has been done on Human Activity Recognition (HAR) from videos [1, 5, 6]. Often for HAR tasks, background of the image does not contain any useful information. The main step in HAR tasks is to localize the human in the entire image. Foreground and background separation techniques can automatically separate the human from rest of the image without the need of drawing manual bounding boxes.

There are two main approaches for performing foreground and background separation. One is to go by a complete image processing perspective where the foreground is detected by performing thresholded interest point tracking, texture recognition, motion estimation, etc. [7–9]. An alternative approach is to pose the problem as matrix decomposition and estimation. The classical Principal Component Analysis (PCA) [10] assumes that the background belongs to a low-rank subspace and is corrupted by additive sparse noise. Through PCA, one estimates the basis of low-rank subspace by maximum variance method.

Suppose for a given matrix M, L is the low rank component and S is the sparse

noise, then M can be decomposed as follows:

$$M = L + S,$$

where,

L - Low-Rank matrix (Background)

S - Sparse matrix (Foreground)

For a given video, background is defined as the part which is either static or has very slow changing environment throughout the video frames. Under this definition, we can model the low-rank matrix L as background. Similarly, foreground is defined as the part that has relatively faster motion and does not occupy huge chunk of the image. Under this definition, the sparse component of the matrix can be modeled as foreground.

So, the problem formulation would be:

$$\min_{L,S} \text{Rank}(L) + ||S||_0 \quad \text{subject to} \quad L + S = M. \tag{1.1}$$

Here, $||\quad||_0$ is the l_0 norm defined as the number of non-zero elements in a given matrix. However (1.1) cannot be solved via convex optimization since finding rank in a convex setting is an NP-hard problem and l_0 norm is non-convex.

The problem formulation also seems ill-posed since the number of unknowns is twice the number of known variables. But fortunately, it is possible to not only estimate the values but also to find exact solution as is proved in [11].

PCA [10] tries to formulate (1.1) slightly differently as follows:

$$\min ||M - L||_F^2 \quad \text{subject to} \quad \text{Rank}(L) \leq k. \tag{1.2}$$

In (1.2), $||\quad||_F$ is the Frobenious norm of the matrix defined as the square root of sum of squares of all elements in a matrix. PCA is widely used as a pre-processing step in many data-processing applications. It gives excellent results when the noise to signal ratio is low. However, if the sparse matrix has arbitrarily large values, PCA fails because Frobenious norm minimization cannot optimally account for gross errors and also does not guarantee sparsity.

To make PCA robust against large but sparse errors, Candes et al. proposed Robust PCA (RPCA) [11]. In RPCA, rank and l_0 norm are replaced by their convex surrogates and then the problem can be solved via convex optimization.

$$\min ||L||_* + \lambda||S||_1 \quad \text{subject to} \quad L + S = M. \tag{1.3}$$

Here, $||\quad||_*$ is the nuclear norm of the matrix, i.e., it is the sum of all singular values, while $||\quad||_1$ is the l_1 norm i.e., it is the sum of the absolute of all entries in the matrix. The above objective function can be solved via Alternating Lagrange Multiplier (ALM) to iteratively optimize for L, S and Y, where Y is the lagrange multiplier. A rigorous proof about the convergence of the algorithm is provided in [11].

RPCA gives impressive results for gross noise removal tasks. However, RPCA is a batch method. That is, RPCA relies on the nuclear norm which requires access to entire dataset in question at once. If there are new additions in the data, the new data has to be appended to the existing samples and RPCA has to be run again. This makes it impractical for real-time applications.

Moreover, the above formulation works under the assumption that the low-rank component is not sparse and sparse component is not low-rank. In applications like background separation, often the sparse component refers to a human or some object which moves slowly over time. This is a very strong example where the sparse component is both sparse and low-rank.

There have been many research ideas to extend RPCA for real-time applications [12–15]. Jiashi Feng et al. [16] proposed online robust PCA via stochastic optimization to overcome the limitation of batch processing. In this paper, the low-rank matrix L is re-written in matrix factorization form ($L = DR^T$). The modified objective function can then be written as follows:

$$\min_{D,R,S} ||M - DR^T - S||_F^2 + \lambda_1\{||D||_F^2 + ||R||_F^2\} + \lambda_2||S||_1 \quad \text{s.t.} \quad DR^T + S = M. \tag{1.4}$$

As can be seen from (1.4), the objective function is not jointly convex with respect to D, R and T. But, we can solve for each of the variables while fixing others. The idea of online matrix factorization and online dictionary update is discussed in [17]. This

modification enables online separation and also adapts for changing backgrounds as the dictionary is updated with every new frame that comes in.

Vaswani et al. [18] also proposed a version of real-time Principal Component Pursuit (PCP) to address slowly changing subspaces and sparse matrices. In parallel, He et al. proposed a robust online separation algorithm named GRASTA [13]. GRASTA will be discussed in detail in Chapter 4.

All the works discussed so far have a common approach of vectorizing the input frames. However, vectorization does not explore the multi-dimensional aspect of the given data. There has been recent interest in exploring the possibilities of the same task in multi-dimensional perspective using tensors. Tensors can be briefly defined as multi-dimensional matrices. A vector is 1D tensor, matrix is 2D tensor and higher dimensional objects are referred to as nD tensors. A detailed introduction to tensors is provided in Chapter 2.

Goldfarb et al. [19] proposed Higher Order Robust PCA (HORPCA) which is an extension of classical RPCA to tensors. Out of many versions of objective functions proposed, singleton model (HORPCA-S) proved to give best results. Let's consider that a given N-th order tensor $\mathcal{M}$ can be written as the sum of a low-rank tensor $\mathcal{L}$ and a sparse tensor $\mathcal{S}$. The objective function can be written as follows:

$$\min_{\mathcal{L},\mathcal{S}}\{\sum_{i=1}^{N}||L^{(i)}||_* + \lambda_1||\mathcal{S}||_1 \quad | \quad \mathcal{L}+\mathcal{S}=\mathcal{M}\}. \tag{1.5}$$

In (1.5), $L^{(i)}$ is the low-rank matrix for the corresponding mode of the tensor. The notations of the tensor algebra and its basics are covered in Chapter 2. The nuclear norm is minimized over all N modes of a tensor. The objective function is solved by Alternating Direction Augmented Lagrange (ADAL). HoRPCA-S proves to have lower error in accuracy when compared with classical RPCA. Zhang et al. [20] proposed an alternative algebra for tensors and proposed Online Tensor Robust PCA (OTRPCA). The details of the algorithm are presented in Chapter 6. As an exploratory exercise, the above approach is extended to mini-batch in this thesis. The proposed mini-batch RPCA speeds up the algorithms while maintaining the accuracy.

Sobral et al. [21] used traditional tensor algebra to propose another online tensor

RPCA algorithm with stochastic optimization. This method is formulated as below:

$$\min_{\mathcal{L},\mathcal{S}} \quad \frac{1}{2}\{\sum_{i=0}^{N} ||\mathcal{M}^{(i)} - \mathcal{L}^{(i)} - \mathcal{S}^{(i)}||_F^2 + \lambda_1||\mathcal{L}^{(i)}||_* + \lambda_2||S^{(i)}||_1\}. \tag{1.6}$$

The above objective function basically tries to optimize for each of the modes of the tensor. It can be solved via ADAL similar to its vector formulation. A lot of other ideas have been explored in implementing higher-order Robust PCA. Interested readers can refer to these works [22–25].

In this thesis, we develop a simple extension of OTRPCA to mini-batch version. This extension significantly improves the computational efficiency of the algorithm while not having to sacrifice accuracy.

We also propose multi-linear GRASTA (MLG), a tensor extension of the GRASTA algorithm proposed in [13]. Detailed experiments with artificial and real datasets show that MLG is more computationally efficient and competent in accuracy compared to other vector and tensor based approaches like [13], [21].

1.1 Thesis Outline

Rest of the thesis is organized as follows. In Chapter 2, the reader is introduced to tensor algebra and other mathematical conventions that will be followed in the rest of the chapters. In Chapter 3, we extend the online Tensor RPCA to mini-batch setting. We discuss results of various experiments to show mini-batch is faster compared to online. Chapter 4 elaborates the GRASTA algorithm. The extension of GRASTA to tensors is discussed in Chapter 5. Chapter 6 mainly deals with numerical experiments to test the performance of MLG. Experiments were done using both artificially generated data and popular background separation datasets like Lobby dataset [26]. Results show that MLG has computational edge over vector-based GRASTA and is competitive in terms of accuracy. Summary and future prospects are discussed in Chapter 7.

Chapter 2

Introduction to Tensors

Tensors are multi-dimensional arrays. 1-d tensors are called vectors, 2-d tensors are matrices and n-d tensors for $n > 2$ are called higher-order tensors. An n-dimensional tensor is indexed by 'n' indices. For example for an n-tensor $\mathcal{A} \in \mathbb{R}^{(I_1 \times I_2 \times \cdots \times I_n)}$, an element 'a' in the tensor $\mathcal{A}$ is denoted by $a_{i_1,i_2,i_3,\ldots,i_n}$. All the definitions in this chapter are taken from [27].

The tensor dimensions are also defined as ways/modes/order. For instance, a 3-d tensor is referred to as 3-way tensor or 3-mode tensor.

Keep in mind, vectors are represented with small letters with an arrow on top ($\overrightarrow{a}$, $\overrightarrow{b}$, etc.) Matrices are represented as simple capital letters (A, B, etc.) Tensors are represented in calligraphic capital letters like ($\mathcal{A}, \mathcal{B}$, etc.) For further discussion, let us assume a 3-tensor $\mathcal{A}$. Now, we introduce the notations and their meaning with respect to tensor $\mathcal{A}$.

Let the dimensions of $\mathcal{A}$ be $i \times j \times k$. Each element in $\mathcal{A}$ is represented as a_{xyz} where x, y and z can range from 0 to $i-1, j-1$ and $k-1$ respectively. $A_{(n)}$ is called the n-th matrix in sequence of tensor $\mathcal{A}$. That is, if we imagine $\mathcal{A}$ to be a stack of 'k' matrices of size $i \times j$, then $A_{(n)}$ is the n-th matrix in the stack where n can range from 1 to k.

Tensor equivalent of matrix rows and columns are called fibers. A fiber is defined by fixing every index but one in a tensor. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $x_{:jk}, x_{i:k}$ and $x_{ij:}$, respectively. When extracted from the tensor, fibers are always assumed to be oriented as column vectors.

Two-dimensional sections of a tensor defined by fixing all but two indices are called slices. The horizontal, lateral, and frontal slices of a third-order tensor $\mathcal{X}$ are denoted

by $X(i ::),\ X(: j :)$, and $X(:: k)$ respectively.

Tensor Frobenious Norm: The Frobenious norm of a tensor $\mathcal{A} \in \mathbb{R}^{i \times j \times k}$ is the square root of sum of squares of all its elements, i.e.,

$$||\mathcal{A}||_F = \sqrt{\sum_{x=0}^{i-1} \sum_{y=0}^{j-1} \sum_{z=0}^{k-1} a_{xyz}^2}.$$

This is analogous to the Frobenious norm of matrices.

Tensor Product: Inner product of tensors is element wise product of two tensors.

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{x=0}^{i-1} \sum_{y=0}^{j-1} \sum_{z=0}^{k-1} a_{xyz} b_{xyz}.$$

We can see that inner product of a tensor with itself is equal to its Frobenious norm.

Tensor Matricization: Elements of any tensor can be re-arranged in many ways into matrix form. This process is called matricization. There are many ways of matricization; the most relevant for this thesis is mode-n matricization.

The mode-n matrix of tensor $\mathcal{A}$ is denoted by $A^{(n)}$. The columns of matrix $A^{(n)}$ are the mode-n fibers of the tensor $\mathcal{A}$. The order of arrangement of columns is not important as long as the arrangement is consistent throughout operations.

Tensor Rank: Tensor $\mathcal{A}$ is called a rank-1 tensor if it can be written as an outer product of 3 vectors. In general, an N-th order tensor is rank-1 if it can be written as an outer product of N vectors.

Rank of the tensor $\mathcal{A}$ is defined as the smallest number of rank-1 tensors that can generate $\mathcal{A}$ as their sum. Finding the rank of tensors is an NP-complete problem [28]. For practical purposes, rank is defined as the number of components of Canonical Polyadic (CP) decomposition that are required for a good reconstruction of the tensor. CP decomposition can be seen as the tensor equivalent for matrix Singular Value Decomposition (SVD).

Chapter 3

Online Tensor Robust PCA

The first contribution of this thesis is an extension of the online tensor robust PCA algorithm to mini-batch setting. There are many versions of tensor RPCA [19], [21], [22]. In this chapter, we explain the online tensor RPCA algorithm and extend it to the mini-batch setting and discuss results.

Below, we have briefly outlined the tensor algebra used in [20]. Detailed proofs can be found in [29]. Discussion of the proofs is out of scope for this thesis.

3.1 Tensor Algebra

Given a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, frames of a video are generally assumed to be stacked as frontal slices of a tensor i.e., we assume that there are n_3 frames of size $n_1 \times n_2$. In the work proposed by Zhang et al. [20], the frames are arranged as lateral slices of $\mathcal{A}$ i.e., there are n_2 frames of size $n_1 \times 1 \times n_3$.

Let $\overrightarrow{v} \in \mathbb{R}^{1 \times 1 \times n_3}$ be called a tubal scalar of tensor $\mathcal{A}$. The t-product (represented by $*$) between two tubal scalars is defined as follows:

$$\overrightarrow{w}(i) = \overrightarrow{u} * \overrightarrow{v} = \sum_{i=0}^{n_3-1} \overrightarrow{u}(k)\overrightarrow{v}((i-k)\text{mod}(n_3)).$$

Given two third-order tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}, \mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$, the t-product ($\mathcal{C} \in \mathbb{R}^{n_1 \times n_4 \times n_3}$) of $\mathcal{A}$ and $\mathcal{B}$ is given as:

$$\mathcal{C}(i,l,:) = \mathcal{A} * \mathcal{B} = \sum_{j=0}^{n_2-1} \mathcal{A}(i,j,:) * \mathcal{B}(j,l,:).$$

Based on the above formulation, tensor-SVD (t-SVD) has been introduced in [29]. For a given tensor $\mathcal{A}$, its t-SVD can be written as $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$ where, $\mathcal{U}$ and $\mathcal{V}$ are

orthogonal tensors and $\mathcal{S}$ is a tensor whose frontal slices are diagonal matrices. The proof can be found in [29].

Based upon these basics, we are now ready to describe the online tensor robust PCA algorithm.

3.2 Online Tensor RPCA

Let's assume a tensor $\mathcal{Z} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ such that there are n_2 frames of size $n_1 \times n_3$. $\mathcal{Z}$ can be decomposed into the sum of a low-rank tensor ($\mathcal{X}$) and a sparse tensor ($\mathcal{E}$) i.e., $\mathcal{Z} = \mathcal{X} + \mathcal{E}$. Given $\mathcal{Z}$, the goal is to estimate $\mathcal{X}$ and $\mathcal{E}$. Mathematically, the objective function can be written as:

$$\min_{\mathcal{X},\mathcal{E}} \frac{1}{2}(||\mathcal{Z} - \mathcal{X} - \mathcal{E}||_F^2 + \lambda_1||\mathcal{X}||_{TNN} + \lambda_2||\mathcal{E}||_1), \tag{3.1}$$

where λ_1 and λ_2 are regularization parameters and $||\mathcal{X}||_{TNN}$ is tensor nuclear norm defined as the sum of singular values of a given tensor. To calculate the tensor nuclear norm, one would need to perform t-SVD on the tensor which requires access to entire dataset at once. Hence, this formulation is not suitable for real-time applications. As a modification, $\mathcal{X}$ can be decomposed into product of a tensor basis ($\mathcal{L}$) and a coefficient tensor ($\mathcal{R}$) i.e., $\mathcal{X}$ can now be written as $\mathcal{X} = \mathcal{L} * \mathcal{R}^T$. Both $\mathcal{L}$ and $\mathcal{R}$ can be estimated and optimized in an online setting. This transformation will alleviate the need for SVD computation. However, it is important to note that the rank 'r' of the tensor has to be defined by user. The algorithm does not have the capability to automatically estimate the optimal rank. This can be done via running t-SVD once on few samples of dataset. By Lemma 3.1 in [20], we can write

$$||\mathcal{X}||_{TNN} = \inf_{\mathcal{L} \in \mathbb{R}^{n_1 \times r \times n_3}, \mathcal{R} \in \mathbb{R}^{n_2 \times r \times n_3}} \left\{ \frac{n_3}{2}(||\mathcal{L}||_F^2 + ||\mathcal{R}||_F^2) : \mathcal{X} = \mathcal{L} * \mathcal{R}^T \right\}. \tag{3.2}$$

Using (3.2), we can re-write (3.1) as follows:

$$\min_{\mathcal{L},\mathcal{R},\mathcal{E}} \frac{1}{2}||\mathcal{Z} - \mathcal{L} * \mathcal{R}^T - \mathcal{E}||_F^2 + \frac{\lambda_1 n_3}{2}(||\mathcal{L}||_F^2 + ||\mathcal{R}||_F^2) + \lambda_2||\mathcal{E}||_1 \quad s.t \quad \mathcal{X} = \mathcal{L} * \mathcal{R}^T. \tag{3.3}$$

The objective function in (3.3) is not jointly convex with respect to all three variables ($\mathcal{L}, \mathcal{R}$ and $\mathcal{E}$). It is convex with respect to each variable when the others are fixed.

Consider sequentially observed data $\{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_{n_2}\} \in \mathbb{R}^{n_1 \times 1 \times n_3}$. Assuming that the dictionary is fixed, the objective function in online setting can be written as follows:

$$\min_{\mathcal{R},\mathcal{E}} \frac{1}{2}||\mathcal{Z}_i - \mathcal{L} * \mathcal{R}^T - \mathcal{E}||_F^2 + \frac{\lambda_1 n_3}{2}||\mathcal{R}||_F^2 + \lambda_2||\mathcal{E}||_1. \tag{3.4}$$

In (3.4), $\mathcal{Z}_i$ is the i-th sequentially observed data. We can solve for $\mathcal{R}$ and $\mathcal{E}$ for each sample through alternating minimization. When $\mathcal{E}$ is fixed, estimating $\mathcal{R}$ reduces to ridge-regression problem which has a closed-form solution.

When $\mathcal{R}$ is fixed, the objective function reduces to LASSO regression problem. $\mathcal{E}$ can now be estimated by soft-thresholding as described in [30]. Below, the soft-thresholding $S_\lambda[x]$ operator is given.

$$S_\lambda[x] = \begin{cases} x - \lambda, & x > \lambda, \\ x + \lambda, & x < -\lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

3.3 Extension to Mini-batch Setting

In this section we will discuss an extension of above formulation to mini-batch setting. In general, batch-mode processing has better performance compared to online processing methods. However, batch mode is much slower compared to online setting. Hence, it is desirable to have an algorithm which is faster than the batch-mode processing and also has better accuracy compared to online processing. Mini-batch implementations usually fall in this trade-off. Mini-batch methods tend to be slower than online but deliver better performance. In mini-batch processing, we process more than one frame at a time. The choice of number of frames to process at once is empirically determined. However, in the mini-batch extension proposed here, we show that the mini-batch processing is both faster and has better performance compared to online method. This is because the dictionary update step in the algorithm is not iterative; it updates only once per sample (sample here can be single frame or multiple-frames).

The mini-batch formulation diverges from the online setting starting from (3.4). Let the size of the mini-batch be 'b' i.e., instead of processing one frame at a time, we

process 'b' frames at a time. Consider the input samples $\{\mathcal{Z}_1, \mathcal{Z}_2, \dots \mathcal{Z}_{n_2/b}\} \in \mathbb{R}^{n_1 \times b \times n_3}$.

$$\min_{\mathcal{R},\mathcal{E}} \frac{1}{2}||\mathcal{Z}_i - \mathcal{L} * \mathcal{R}^T - \mathcal{E}||_F^2 + \frac{\lambda_1 n_3}{2}||\mathcal{R}||_F^2 + \lambda_2||\mathcal{E}||_1, \tag{3.6}$$

where $\mathcal{R} \in \mathbb{R}^{r \times b \times n_3}$ and $\mathcal{E} \in \mathbb{R}^{n_1 \times b \times n_3}$

Equation (3.6) can be solved for $\mathcal{R}$ by ridge-regression while keeping $\mathcal{E}$ fixed. $\mathcal{E}$ can be estimated via soft-thresholding when $\mathcal{R}$ is fixed. The dictionary or the spanning tensor basis $\mathcal{L}$ is updated as described in Algorithm 3. The proof of online dictionary update can be found in [17]. The mini-batch extension algorithm is outlined below. It is known that convolution in time domain is equivalent to multiplication in Fourier domain. Since t-product is nothing but circular convolution, it is performed via Fourier domain multiplications in the following algorithm. This transformation is more computationally efficient than performing circular convolution. Tranformations to Fourier domain and back are done through Fast Fourier Transformation (fft) and Inverse fft (ifft). Note that fft of a tensor $\mathcal{A}$ along its third dimension is denoted by $\text{fft}(\mathcal{A}, [], 3)$ in the following algorithms.

Mini-Batch Tensor Robust PCA (MBTRPCA) Algorithm:

Input: Mini-batch samples $\mathcal{Z}_t \in \mathbb{R}^{n_1 \times b \times n_3}$.

Initialize: $\mathcal{L}_0 \in \mathbb{R}^{n_1 \times r \times n_3}, \mathcal{R}_0 \in \mathbb{R}^{r \times b \times n_3}, \mathcal{E}_0 \in \mathbb{R}^{n_1 \times b \times n_3}$

1. **for** each mini-batch sample
2. Find $\mathcal{R}_t, \mathcal{E}_t$ using Algorithm 2.
3. $\{\mathcal{R}_t, \mathcal{E}_t\} = \min_{\mathcal{R},\mathcal{E}} \frac{1}{2}||\mathcal{Z}_t - \mathcal{L} * \mathcal{R}^T - \mathcal{E}||_F^2 + \lambda_1 n_3/2||\mathcal{R}||_F^2 + \lambda_2||\mathcal{E}||_1$
4. Compute $\mathcal{A}_t = \mathcal{A}_{t-1} + \mathcal{R}_t * \mathcal{R}_t^T$, $\mathcal{B}_t = \mathcal{B}_{t-1} + (\mathcal{Z}_t - \mathcal{E}_t) * \mathcal{R}_t^T$
5. Update spanning Tensor basis $\mathcal{L}_t$ using Algorithm 3.

$$\mathcal{L}_t = \min_{\mathcal{L}} \frac{1}{2}\text{tr}[\mathcal{L}^T(\mathcal{A}_t + n_3\lambda_1\mathbf{I})\mathcal{L}] - \text{tr}(\mathcal{L}^T\mathcal{B}_t)$$

6. end **for**
7. **Return** $\mathcal{X} = \mathcal{L} * \mathcal{R}^T$ and $\mathcal{E}$

Algorithm 2: Find $\mathcal{R}_t$ and $\mathcal{E}_t$

1. $\hat{z}_t$ = fft ($\mathcal{Z}_t$,[],3), $\hat{L}_{t-1}$ = fft ($\mathcal{L}_{t-1}$,[],3)

2. **while** not converge **do**

3. **for** i = 1,2,... n_3 **do**

4. $\hat{r}_t^{(i)} = ((\hat{\mathcal{L}}_{t-1}^{(i)})^T(\hat{\mathcal{L}}_{t-1}^{(i)}) + \lambda_1 I)^{-1}(\hat{\mathcal{L}}_{t-1}^{(i)})^T(\hat{z}_t^{(i)})$

5. end **for**

6. $\mathcal{R}_t$ = ifft($\hat{r}_t$,[],3).

7. $\mathcal{E}_t = S_{\lambda_2}[\mathcal{Z}_t - \mathcal{L}_t i 1^T * \mathcal{R}_t]$

8. end **while**

9. **Return** $\mathcal{R}_t, \mathcal{E}_t$.

Algorithm 3: Tensor basis update

1. $\mathcal{L}_t = \mathcal{L}_{t-1}$, $\hat{\mathcal{L}}_t$ = fft($\mathcal{L}_t$,[],3)

2. $\hat{\mathcal{B}}_t$ = fft($\mathcal{B}_t$,[],3)

3.$\mathcal{C}_t = \mathcal{A}_t + \lambda_1 \mathcal{I}$, $\hat{\mathcal{C}}_t$ = fft($\mathcal{C}_t$,[],3)

4.**for** j= 1,2,... r **do**

5. **for** k=1,2,... n_3 **do**

6. $\hat{\mathcal{L}}_t(:,j,k) = \frac{\hat{\mathcal{B}}_t(:,j,k) - \hat{\mathcal{L}}_t(:,:,k) * \hat{\mathcal{C}}_t(:,j,k)}{\hat{\mathcal{C}}_t(j,j,k)} + \hat{\mathcal{L}}_t(:,j,k)$

7. end **for**

8. end **for**

9. **Return** $\mathcal{L}_t$ = ifft($\hat{\mathcal{L}}_t$,[],3).

3.4 Results and Discussion

To validate the algorithm, an artificial data set of 500 frames of size 100 × 100 was generated as the sum of a low-rank tensor and a sparse tensor. The entries of the low-rank tensor are i.i.d. random Gaussian $\mathcal{N}(0,1)$. Entries of the sparse tensor are i.i.d. random Gaussian $\mathcal{N}(0,10)$. The rank was pre-defined as 5. The error metric used for the estimation of low-rank component is Relative Squared Error (RSE). RSE is defined as following:

$$\text{RSE}(t) = \frac{||\bar{\mathcal{L}}_t - \mathcal{G}_t||_F}{||\mathcal{G}_t||_F},$$

where $\bar{\mathcal{L}}_t$ is the estimated low-rank component and $\mathcal{G}_t$ is the ground truth.

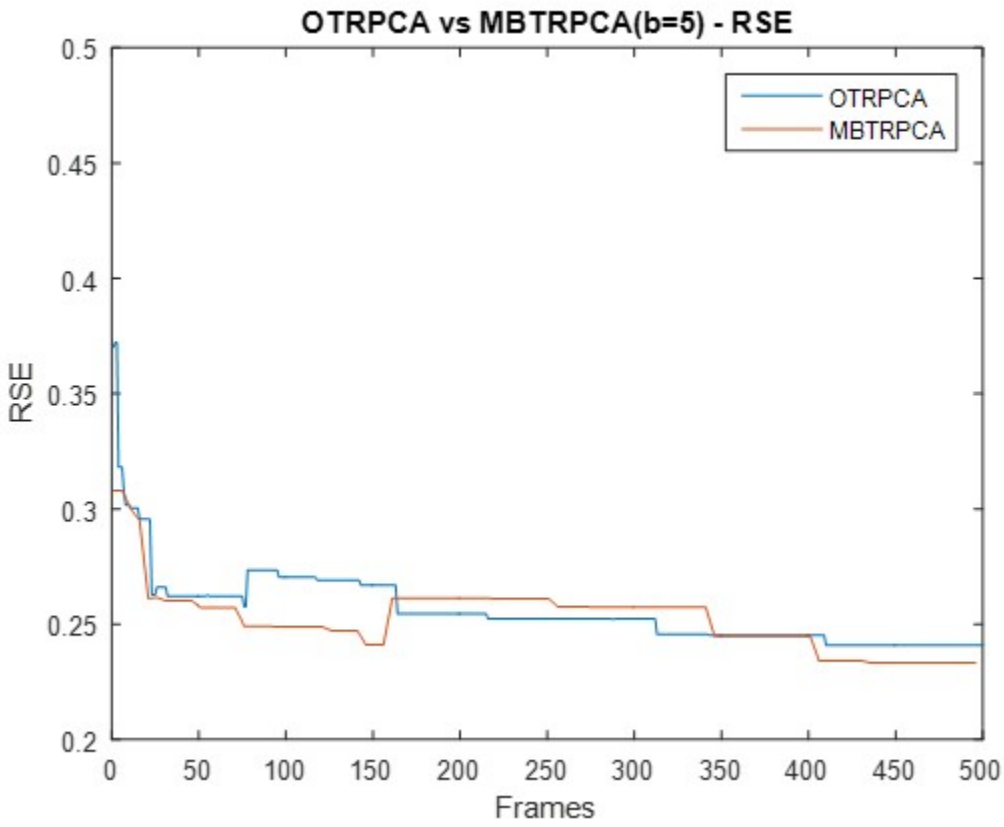

Figure 3.1: RSE of mini-batch (b=5) converges almost similar to Online Tensor RPCA. The mini-batch implementation is 3.75 times faster than the online implementation.

RSE is measured for every input sample i.e., in the case of OTRPCA, RSE is measured after for every frame while in the case of mini-batch one RSE value corresponds to the set of 'b' frames taken as input. Figure 3.1 shows the RSE curve of OTRPCA and MBTRPCA ($b = 5$) across 500 frames. We notice that mini-batch convergence is very similar to OTRPCA.

A quick re-visit to Algorithm3 shows that the subspace update is independent of the mini-batch size 'b'. This means that for OTRPCA the subspace update takes place after every frame, while for MBTRPCA the subspace update happens only every 'b' frames. Hence, MBTRPCA is much faster than OTRPCA. Moreover, since the dictionary has access to more than one frame at a time, the dictionary update also converges faster in MBTRPCA as is seen in Figure 3.1.

Next, we test both the algorithms on the basketball video from [29]. This video has 40 frames of size 144×256. To these frames, i.i.d. random Gaussian noise was added

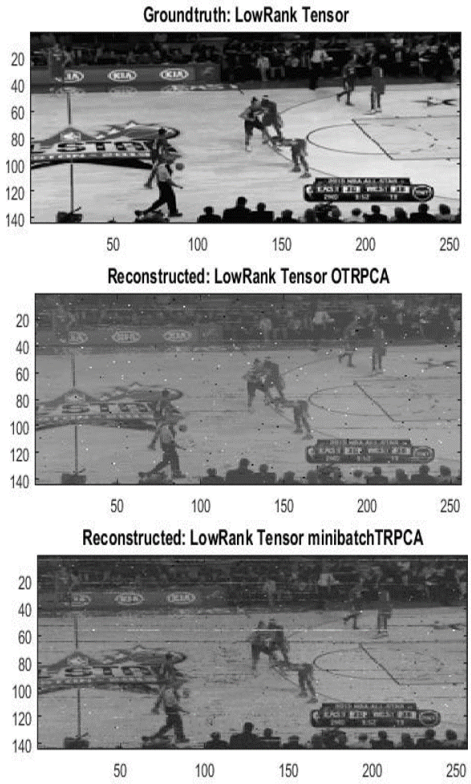

Figure 3.2: Low-rank reconstruction of basketball video using OTRPCA and MBTRPCA. The first subplot shows the ground truth and next subplots show the reconstruction by OTRPCA and MBTRPCA.

in pre-determined sparse locations. The idea is that the algorithms should successfully separate the sparse noise while preserving the low-rank background information. The results can be seen in Figure 3.2. We can see that MBTRPCA has separated out sparse noise better than OTRPCA.

A total of 1000 Monte-Carlo simulations were run on both algorithms using the basketball video data set. Table 3.1 shows the RSE values of OTRPCA and MBTRPCA for five iterations where each iteration is one run of Monte-Carlo. We observe that the RSE of MBTRPCA is always lower than OTRPCA. Table 3.2 shows the amount of time taken to separate 40 frames. As can be seen from the table, MBTRPCA is approximately 3.7 times faster than OTRPCA for a mini-batch size of 5.

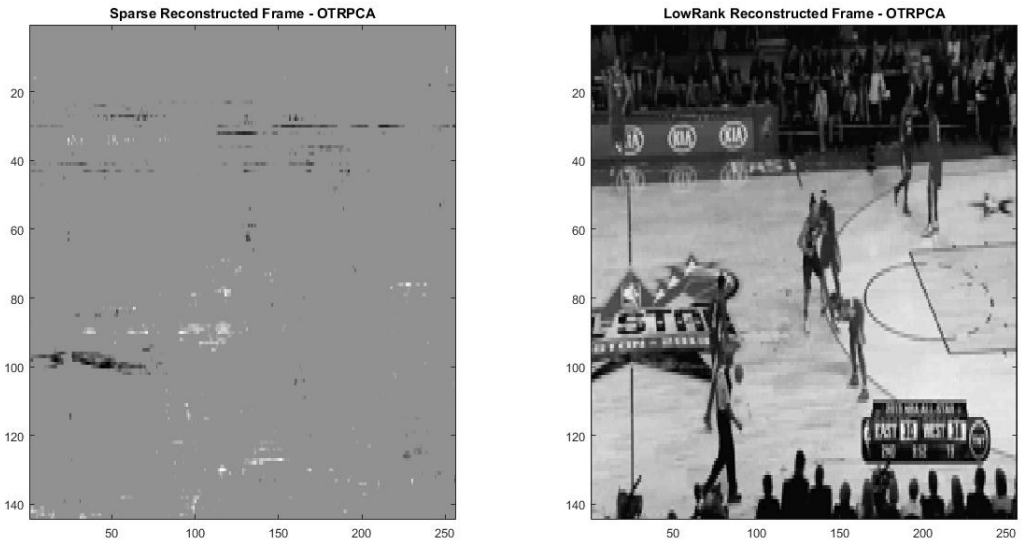

Figure 3.3: Sparse and low-rank decomposition of basketball video without any additive sparse noise using OTRPCA. OTRPCA does not succeed in separating humans as sparse foreground.

Method/RSE	Iter1	Iter2	Iter3	Iter4	Iter5	Avg RSE
OTRPCA	0.4306	0.4662	0.4648	0.4304	0.4326	0.445
MBTRPCA	0.3849	0.3581	0.3922	0.3869	0.4020	0.384

Table 3.1: RSE for first 5 iterations of Monte-Carlo simulation

Next, we explore if the algorithm would succeed in separating out humans from the video when no other noise is added. We can see the results in Figure 3.3 that OTRPCA does not succeed. This is mainly because both the algorithms do not have any background training phase. Also, the dictionary update algorithm as discussed previously is non-iterative, In the next chapter, we discuss an algorithm GRASTA which overcomes the limitations of the algorithms discussed in this chapter.

Method/Time(sec)	Iter1	Iter2	Iter3	Iter4	Iter5	Avg time
OTRPCA	275.9967	280.9400	260.1825	294.9070	280.3871	278.4826
MBTRPCA	90.3468	87.9617	81.6807	91.4076	85.7323	87.426

Table 3.2: Computational time for first 5 iterations of Monte-Carlo simulation

Chapter 4

Vector-based Grassmanian Robust Adaptive Subspace Tracking Algorithm (GRASTA)

In this chapter we discuss the background separation algorithm GRASTA [13] and form basics for the next chapter where we extend GRASTA to tensor-setting.

Assume at time 't', a vectorized image frame $\overrightarrow{v_t}$ arrives. $\overrightarrow{v_t}$ can be decomposed as the sum of a low-rank vector ($\overrightarrow{l_t}$) and a sparse vector ($\overrightarrow{s_t}$). $\overrightarrow{l_t}$ can further be decomposed as the product of a low-rank subspace basis and a coefficient vector i.e., $\overrightarrow{l_t} = U_t\overrightarrow{w_t}^T$. $\overrightarrow{v_t}$ can be written as follows:

$$\overrightarrow{v_t} = U_t\overrightarrow{w_t}^T + \overrightarrow{s_t}. \tag{4.1}$$

Given $\overrightarrow{v_t}$, the task is to estimate U_t, $\overrightarrow{w_t}$ and $\overrightarrow{s_t}$. The loss function for quantifying subspace error is defined as

$$F(S,t) = \min_{w} ||U_t w_t^T - v_t||_1. \tag{4.2}$$

$||\quad||_1$ is the l_1 norm defined as the sum of absolute values of a vector. If U_t is known or estimated, then (4.1) can be solved via Alternating Directions Method of Multipliers (ADMM). The above loss function can alternatively be written in the augmented Lagrangian form as follows:

$$\mathcal{L}(U,w,s,y) = ||\overrightarrow{s}||_1 + \overrightarrow{y}^T(U_t\overrightarrow{w} + \overrightarrow{s} - \overrightarrow{v}) + \frac{\rho}{2}||U_t\overrightarrow{w} + \overrightarrow{s} - \overrightarrow{v}||_2^2, \tag{4.3}$$

where ρ is the regularization parameter and $\overrightarrow{y} \in \mathbb{R}^n$ is the dual vector. The above expression is not jointly convex with respect to all the variables. However, if we assume U_t is known or fixed, then rest of the variables can be easily updated using ADMM in the following way. If all other variables except $\overrightarrow{w_t}$ are fixed, then the problem reduces to a ridge regression problem. Ridge regression has a closed form solution as long as

the covariance matrix $U_t^T U_t$ is invertible.

$$\begin{aligned} w^{k+1} &= \frac{1}{\rho}(\hat{U_t}^T \hat{U_t})^{-1} \hat{U}^T(\rho(v_t - s^k) - y^k), \\ s^{k+1} &= ST_{\frac{1}{1+\rho}}(v_t - \hat{U}w^{k+1} - y^k), \text{ and} \\ y^{k+1} &= y^k + \rho(\hat{U}w^{k+1} + s^{k+1} - v_t), \end{aligned} \tag{4.4}$$

where $ST_{\frac{1}{1+\rho}}$ is the soft thresholding operator. The augmented Lagrangian minimization (4.4) only works when U_t is known. Hence, U_t has to be updated separately. To this end, GRASTA proposes Grassmanian subspace update. Grassmanian is the set of all linear subspaces of fixed dimension 'd'. It can be concisely represented as $G(d, n)$ or $G(d, V)$ where 'n' is the ambient dimension of the vector space V. When vector space V is real or complex, the Grassmanian is a smooth manifold. Hence it has a continuous gradient. When a gradient step of length η is taken along the geodesic of Grassmanian, we end up with the following for subspace update:

$$U(\eta) = U + (\cos(\eta\sigma) - 1)\frac{Uw_t^*}{||w_t^*||_F}\frac{(w_t^*)^T}{||w_t^*||_F} - \sin(\eta\sigma)\frac{\Gamma}{||\Gamma||_F}\frac{(w_t^*)^T}{||w_t^*||_F},$$

where $\Gamma = \big(\overrightarrow{y_t}^* + \rho(U_t\overrightarrow{w_t}^* + \overrightarrow{s_t}^* - \overrightarrow{v_t})\big)(1 - U_t U_t^T)$. Detailed proof for this expression can be seen in [13]. GRASTA is currently the state-of-the-art for background and foreground separation. However, GRASTA dos not consider the multi-dimensional information from the videos. In the next chapter, we try to incorporate multi-dimensional information by extending GRASTA to tensor setting.

Chapter 5

Multi-Linear GRASTA (MLG)

Let us rewrite the problem formulation from previous chapter in tensor setting : $\mathcal{M} = \mathcal{L} + \mathcal{S}$. Here, $\mathcal{M}$ is the n-th order observation tensor, while $\mathcal{L}$ and $\mathcal{S}$ are low-rank and sparse tensors, respectively.

Suppose at time 't', an image frame $\mathcal{M}_t$ arrives. From the above formulation, $\mathcal{M}_t$ can be decomposed as follows

$$\mathcal{M}_t = \mathcal{L}_t + \mathcal{S}_t,$$
$$\mathcal{M}_t = \mathcal{U}_t.\mathcal{W}_t^T + \mathcal{S}_t,$$

where $i = 0, 1, \ldots n$ are the number of modes of the tensor, $\mathcal{U}_t$ is the low-rank subspace, $\mathcal{W}_t$ is the coefficient tensor, and $v_t^{(i)}$ is the vectorized i^{th} mode tensor. For each mode, if $U_t^{(i)}$ is known or estimated, then the above problem can be solved via ADMM. The augmented Lagrangian form of the objective function is written as follows. For each mode i,

$$\mathcal{L}(U^{(i)}, w^{(i)}, s^{(i)}, y^{(i)}) = ||s^{(i)}||_1 + (y^{(i)})^T(U_t^{(i)}.w^{(i)} + s^{(i)} - v^{(i)}) + \frac{\rho}{2}||U_t^{(i)}w^{(i)} + s^{(i)} - v^{(i)}||_2^2, \tag{5.1}$$

where $y^{(i)} \in \mathbb{R}^n$ is the dual vector. The above expression is not jointly convex with respect to all variables. However, if we assume $U_t^{(i)}$ is known or fixed, then rest of the variables can be easily updated using ADMM in following way. When all other variables except $w^{(i)}$ are fixed, problem reduces to ridge regression and we can estimate

w as long as $(U_t^{(i)})^T U_t^{(i)}$ is assumed to be invertible:

$$w^{k+1(i)} = \frac{1}{\rho}(\hat{U_t}^{(i)T}.\hat{U_t}^{(i)})^{-1}\hat{U^{(i)}}^T(\rho(v_t^{(i)} - s^{(i)k}) - y^{(i)k}),$$
$$s^{k+1(i)} = ST_{\frac{1}{1+\rho}}(v_t^{(i)} - \hat{U}^{(i)}w^{k+1(i)} - y^{(i)k}), and$$
$$y^{k+1(i)} = y^{(i)k} + \rho(\hat{U}^{(i)}w^{k+1(i)} + s^{k+1(i)} - v_t^{(i)}),$$

where $ST_{\frac{1}{1+\rho}}$ is the soft thresholding operator. The augmented Lagrangian minimization only works when U_t is known. Hence, U_t has to be updated separately.

For a gradient step of length η along the geodesic of grassmanian, the updated subspace for each mode can be written as follows :

$$U^{(i)}(\eta) = U^{(i)} + (\cos(\eta\sigma) - 1)\frac{U^{(i)}.w_t^{(i)*}}{||w_t^{(i)*}||}\frac{(w_t^{(i)*})^T}{||w_t^{(i)*}||} - \sin(\eta\sigma)\frac{\Gamma}{||\Gamma||}\frac{(w_t^{(i)*})^T}{||w_t^{(i)*}||}.$$

The results are discussed in next chapter. MLG can actually be used to process single or multiple frames at once. We show in the next chapter that processing mutiple-frames at once improves the accuracy as well as computational efficiency as compared to vector-based GRASTA.

Chapter 6

Results and Discussion

This chapter discusses various experiments done with multi-linear GRASTA and its results. There were a series of experiments performed with the algorithm which all revealed various advantages of the tensor setting.

Majority of the algorithms for foreground and background separation only consider gray-scale images as input. Since MLG considers multi-dimensional input, we explore if retaining all channel information would help in improving the foreground and background separation. For this, we take one frame at a time but as an RGB tensor instead of a gray-scale matrix. For the experiments, we work with the Lobby video dataset [13]. The video consists of an indoor static environment where occasionally humans pass by. Video consists of 1546 frames of size 144×256. Figure 6.1 shows the background and foreground separation of RGB image using MLG. While the separation is successful, it is observed that the accuracy of estimation has not improved as compared to using just the gray-scale images for separation. The ground truth for foreground was available in thresholded format i.e., all the foreground pixels correspond to 1 and background pixels are 0. We try to compare the performance of MLG with other algorithms by measuring error against this ground truth. For this, we perform hard thresholding on the estimated foreground by forcing pixels less than a chosen threshold to be 0. We measure pixel-wise error for the foreground. True positives correspond to the number of foreground pixels that match the ground truth. False positives are the number of pixels that are wrongly estimated as foreground. False negatives are the number of pixels that are wrongly estimated as background. Error is now defined as $\text{Error} = \frac{\text{no. of false pos + no. of false negs}}{\text{total no. of pixels in image}}$.

MLG can process multiple frames at a time, which means that we can use MLG in both online and mini-batch setting. We conduct experiments by varying the order of

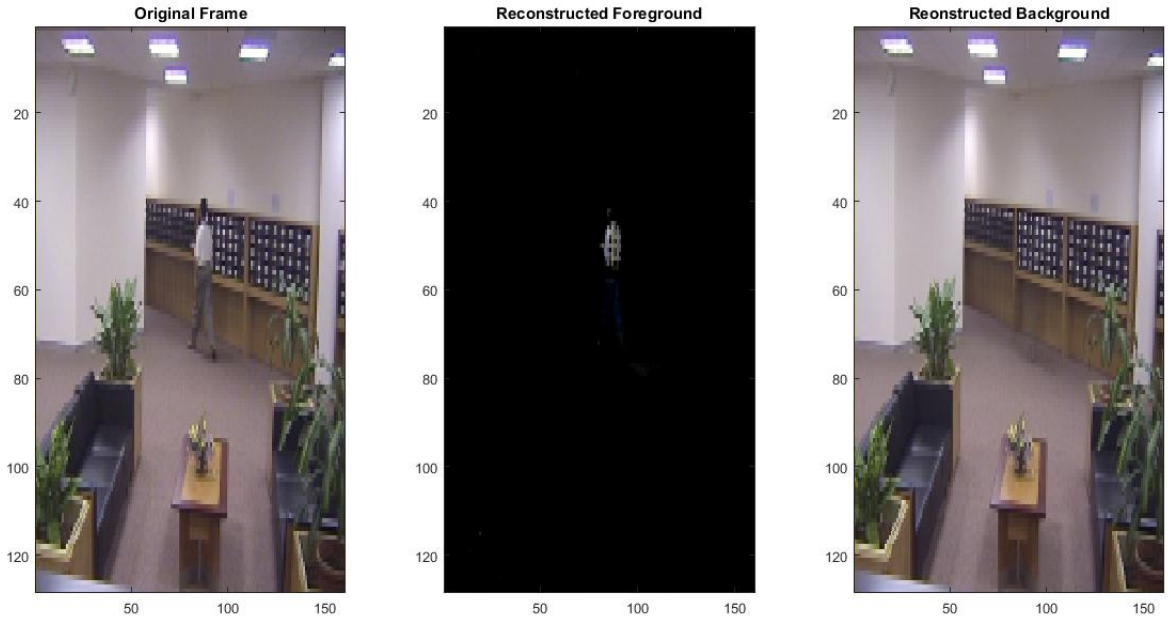

Figure 6.1: MLG on RGB tensor. The left most image is the input to MLG. The second subplot shows the reconstructed foreground and the last image is the reconstructed background using MLG.

the tensor (n) as well as the batch size (b) given as input to MLG. Table 6.1 compares vector-based GRASTA [13] and Online Stochastic Tensor Decomposition (OSTD) [21] algorithms with MLG in terms of performance and computational time for foreground and background separation on Lobby dataset. For all the algorithms, time per frame is calculated as the ratio of total-time taken for processing all frames and the total no. of frames i.e., Time per frame $= \frac{\text{total time for all frames}}{\text{total no. of frames}}$. All experiments were coded in MATLAB and tested on an Intel core i5 with 1.7GHz clock speed and 64GB RAM. As can be seen from the table, MLG in mini-batch setting is the fastest amongst other algorithms. It is worthwhile to notice that none of the GRASTA versions actually have any false positives in estimation.

Figure 6.2 shows the gray-scale reconstruction of MLG performed with a mini-batch size of 8. We see that the reconstruction is very accurate even with a relatively big mini-batch size. The Lobby video has dynamic lighting condition in the video frames. The lighting of video changes over time. When the lighting changes, MLG takes about 30 frames to get adjusted to the new background which is similar to vector-GRASTA.

Method	Error	False Pos	False neg	Time per frame (sec)
GRASTA [13]	0.0128	0	257	0.6752
MLG(b=1)	0.0128	0	257	0.7
MLG (b=5)	0.0132	0	271	0.01
OSTD [21]	0.0164	84	173	0.03

Table 6.1: Time per frame and error for foreground separation on Lobby dataset. MLG ($b = 1$) corresponds to processing one frame at a time, MLG ($b = 5$) corresponds to processing 5 frames at a time.

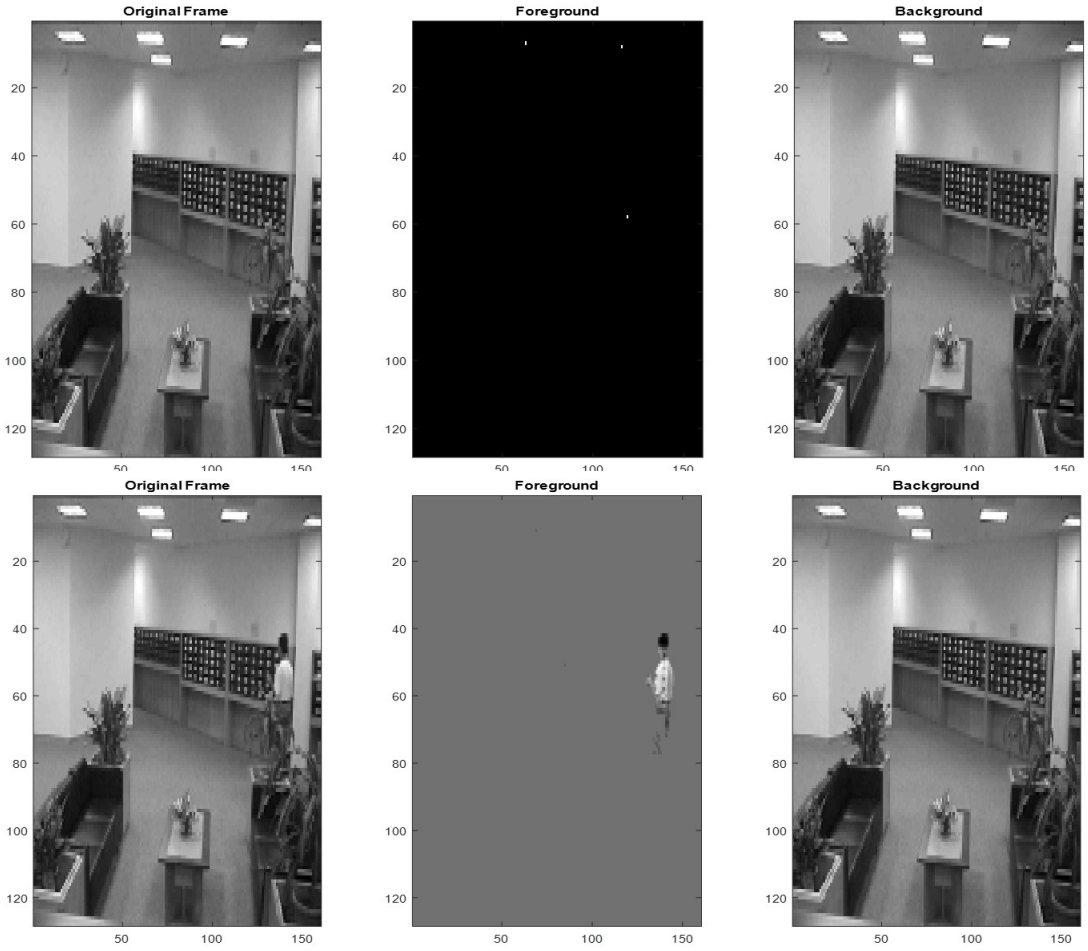

Figure 6.2: Lobby Dataset Sparse and Low-Rank reconstruction by MLG ($b = 8$). The plots from left to right show input image, estimated foreground and background respectively. The first row shows the reconstruction when there is no foreground and the second row shows successful separation of foreground when human is present in the frame.

Chapter 7

Conclusion and Future Work

In the first part of this thesis, we have developed a mini-batch extension to OTRPCA. We show that the extension helps improve performance as well as computational efficiency. However, we also notice that the error in low-rank subspace estimation is pretty high. This could be because there is no background training phase initially. One of the future directions would be to incorporate background training to improve the results.

In the second part of the thesis, we extend vector-based GRASTA to multi-linear setting. Two main pre-requisites of GRASTA and MLG are that the rank must be pre-defined and the background must be pre-trained. A possible future work could be to explore ways of smartly initializing the background with minimal or no training required. That can further reduce the computational cost.

It would also be interesting to incorporate block sparsity or similar structured constraints on sparse tensors to get better separation results. Since MLG is capable of processing multi-channel data, one can also explore using multi-spectral channel data for improved separation using MLG.

References

[1] Tong Wu, Prudhvi Gurram, Raghuveer M Rao, and Waheed U Bajwa. Hierarchical union-of-subspaces model for human activity summarization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–9, 2015.

[2] Xichen Sun, Liwei Wang, and Jufu Feng. Further results on the subspace distance. *Pattern recognition*, 40(1):328–329, 2007.

[3] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

[4] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011.

[5] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, pages 124–1. BMVA Press, 2009.

[6] Gang Yu, Junsong Yuan, and Zicheng Liu. *Unsupervised Trees for Human Action Search*, pages 29–56. Springer Singapore, Singapore, 2015.

[7] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the 14th International Conference on Computer Communications and Networks*, ICCCN '05, pages 65–72, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Christian Thurau and Vclav Hlavc. Pose primitive based human action recognition in videos or still images. In *CVPR*. IEEE Computer Society, 2008.

[9] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[10] Ian Jolliffe. *Principal component analysis*. Wiley Online Library.

[11] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011.

[12] Jieping Ye, Ravi Janardan, and Qi Li. Gpca: an efficient dimension reduction scheme for image compression and retrieval. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 354–363. ACM, 2004.

[13] Jun He, Laura Balzano, and Arthur Szlam. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1568–1575. IEEE, 2012.

[14] Wei Lu and Namrata Vaswani. Modified compressive sensing for real-time dynamic mr imaging. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3045–3048. IEEE, 2009.

[15] Alp Ozdemir and Selin Aviyente. Online low-rank+ sparse structure learning for dynamic network tracking. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4074–4078. IEEE, 2016.

[16] Jiashi Feng, Huan Xu, and Shuicheng Yan. Online robust pca via stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 404–412, 2013.

[17] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[18] Chenlu Qiu and Namrata Vaswani. Real-time robust principal components' pursuit. pages 591–598, 2010.

[19] Donald Goldfarb and Zhiwei Qin. Robust low-rank tensor recovery: Models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1):225–253, 2014.

[20] Zemin Zhang, Dehong Liu, Shuchin Aeron, and Anthony Vetro. An online tensor robust pca algorithm for sequential 2d data. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2434–2438. IEEE, 2016.

[21] Andrews Sobral, Sajid Javed, Soon Ki Jung, Thierry Bouwmans, and El-hadi Zahzah. Online stochastic tensor decomposition for background subtraction in multispectral video sequences. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 106–113, 2015.

[22] Arash Golibagh Mahyari and Selin Aviyente. Identification of dynamic functional brain network states through tensor decomposition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2099–2103. IEEE, 2014.

[23] Emily Miller, Scott Ladenheim, and Carla Martin. Higher order tensor operations and their applications.

[24] Karen Braman. Third-order tensors as linear operators on a space of matrices. *Linear Algebra and its Applications*, 433(7):1241–1253, 2010.

[25] Nicholas D Sidiropoulos and Anastasios Kyrillidis. Multi-way compressed sensing for sparse low-rank tensors. *IEEE Signal Processing Letters*, 19(11):757–760, 2012.

[26] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004.

[27] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[28] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[29] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-svd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3842–3849, 2014.

[30] David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.