Nonnegative Robust PCA for Background and Foreground Image Decomposition

Chenyang Ling

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Nonnegative Robust PCA for Background and Foreground Image Decomposition

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Chenyang Ling

September 2, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Nonnegative Robust PCA for Background and Foreground Image Decomposition

by

Chenyang Ling

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: September 2, 2020

Supervisor(s): dr.ir.K. Batselier

Reader(s): Prof.dr.ir. Jan-Willem van Wingerden

dr. ing. Raf van de Plas

Abstract

Nowadays, video surveillance and motion detection system are widely used in various environments. With the relatively low-price cameras and highly automated monitoring system, video and image analysis on road, highway and skies becomes realistic. The key process in the analysis is to separate the useful information such as moving foreground objects from the original video sequence where Robust Principal Component Analysis (RPCA) plays an important role in extracting the foreground objects. RPCA have been widely used in data analysis and dimension reduction with applications in image recovery, information clustering and computer vision. But one drawback of RPCA lies in the fact that it does not guarantee the nonnegativity of pixels. It is important to have nonnegative foreground object since negative pixels that are not in the range between 0 and 255 are meaningless and the foreground objects are thus not visible. State-of-the-art methods do not consider the nonnegativity of the foreground object in their algorithms.

This thesis focuses on the problem of extracting foreground moving object from background scenes and guarantee the nonnegativity of foreground object. This thesis proposes a method that combines RPCA and Nonnegative Matrix Factorization (NMF). It ensures the pixels that constitute the foreground object is nonnegative by using the basic model of RPCA and nonnegative components that NMF provides. The efficacy of the proposed algorithms is tested on publicly available dataset. Experiment shows in detail how the proposed algorithms achieve in recovering the foreground object with high true positive rate. Together with RPCA algorithm, the performance of recovery is compared and their advantages and disadvantages are discussed.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor dr.ir.K. Batselier for his assistance during the writing of this thesis. We had our meeting both in person and online every two weeks. He always inspired me with new ideas and gave me feasible advice for my algorithm and thesis writing. His guidance throughout the 9 months not only equipped me with sufficient knowledge and ideas for my thesis but also made me dig into the topic of RPCA with more enthusiasm than before.
I would also like to thank my family and friends for giving me consistent mental support during the tough time of lockdown this year. They are like beacon that brought light to my life of writing thesis at home. It is with their encouragement that I can write my thesis and spend everyday life with positivity.
Last but not least, I would like to thank every classmate, colleague and teacher that I met here in TU Delft. Acclimatising the fast-tempo study in the first year was not that easy. With your generous help for my coursework and beneficial advice in life I can spend this 2-year journey with less barrier but more confidence and love. This thesis means the end of my student life at this stage but it is also a brand new start for the rest of my life. I hope I can always have an adventurous heart and curious mind towards everything in the near future like I did in this thesis.

Delft, University of Technology Chenyang Ling
September 2, 2020

Chapter 1

Introduction

In a set of image sequence or video sequence, there are different types of information contained in each frame. Some information can be static such as buildings, mountains and roads. While other information is dynamic such as cars on the road, pedestrians and birds flying in the sky. Besides static and moving objects, there are also some unwanted information like occlusion, weather condition, illumination, etc. When analyzing an image or video sequence, only part of the information is needed. Thus it is important to separate those objects from the original image or video frame. The separation process can be helpful in applications like video surveillance system and motion capture. These applications will use the result of image or video decomposition to analyze the overall environment or the motion of specific object in the image sequence. For example, in [1] the motion and speed of vehicles on highway are monitored by extracting these vehicles from video frames to avoid collision and accidents on highways. In [4] and [5], moving objects are separated from the original video frames to analyse the motion of these objects and identify the behaviour of individuals in various environments. Similarly, the moving objects in the sky are detected in [6] to guarantee the aviation safety.

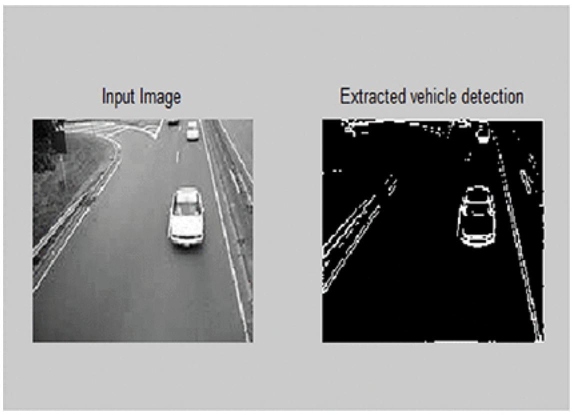

Figure 1-1: Vehicle extraction on highways [1]

And in these applications, a common algorithm Robust Principal Component Analysis (RPCA) is adopted to extract the foreground object from background environment. RPCA decomposes the original image into two parts. One only contains background scenes and the other only contains foreground objects. But RPCA does not restrict the pixels that constitute the foreground object in the range of 0 to 255. One problem it induces is that those negative pixels are meaningless and cannot be shown in images. It means those foreground objects in [4], [5] and [6] are not visible in images. The decomposition result of RPCA is thus futile. One feasible solution is to turn all the negative pixels to zero or small positive values. Then the pixels are within the pixel range again. But the result can lose information of the foreground objects since zero or small value pixels are dark-colored.
To deal with the negative pixels that RPCA brings in a better way, this thesis proposes a new algorithm that is able to extract the foreground object in a set of image sequence in a nonnegative way. On the basis of RPCA, it takes the negative pixels into account and force them to be nonnegative with the help of Nonnegative Matrix Factorization (NMF). The main contribution of the new algorithm is that it turns the negative pixels into positive and the information contained in these pixels are visible again. These two algorithms RPCA and NMF together with their applications in background and foreground image decomposition will be introduced respectively in the next section.

1-1 Introduction to RPCA

The name of RPCA originates from the name PCA. Principal Component Analysis (PCA) was invented in 1901 by Karl Pearson [7] and is used to transform high-dimensional data to lower dimensions using a new coordinate system. In the new coordinate system, the first coordinate which is called the first principal component has the highest variance of the data projection, which means the data varies from each other to the greatest extent in this coordinate. The second coordinate has the second highest variance and so on. PCA is widely used as a tool in exploratory data analysis and for making predictive models [8].
Based on the idea of PCA that reduces the dimensionality of the data matrix, Robust Principal Component Analysis (RPCA) was proposed. It was first proposed in paper [9]. It is a modification of the original PCA problem which aims to recover the low-rank component $L \in R^{m\times n}$ from corrupted observation data matrix $D \in R^{m\times n}$. Here m and n are the dimension of the data matrix. The basic idea in this paper is to decompose the data matrix into two components: low-rank component $L \in R^{m\times n}$ and sparse component $S \in R^{m\times n}$. Then it forms the optimization problem by using Principal Component Pursuit (PCP). This optimization problem can then be solved by algorithms such as Augmented Lagrange Multiplier Method (ALM) and Alternating Direction Method (ADM). The decomposition problem is concluded in Eq. (1-1),

$$D = L + S. \tag{1-1}$$

In some cases, L needs to be recovered from the corrupted data observation matrix D, such as the denoising of images where a clear image will be recovered from speckled data matrix. Sparse component in this case represents image noises. In other cases like the application that is introduced at the beginning of this chapter, the focus is on S. In this case sparse component represents the foreground object which needs to be separated from original image. These two components are both of importance depending on the need of actual application.

In the next section the general applications of PRCA will be introduced. The application of background and foreground separation will be mentioned in detail.

1-2 Applications of RPCA

In this section, more applications of foreground and background decomposition will be introduced. Besides, general applications of RPCA will be briefly introduced to show RPCA is also applicable in other fields.

1-2-1 Background and foreground decomposition applications of RPCA

In this subsection the application of background and foreground decomposition will be explained in detail since in chapter 3 the proposed algorithm will be tested on this application. Similar to the example at the beginning of this chapter, the low-rank component represents the background environment and sparse component is the foreground object.

- **Video surveillance:** This application is widely used in different situations. It can facilitate automated surveillance system to detect the cars on road and ensure the traffic safety. When it comes to the environment of sea, it can detect the vessels on the sea surface. In [3] a new model of RPCA is proposed to extract the foreground moving objects on the sea. It first finds the region of interest in the image then separate it from the background. Figure 1-3 shows the diagram of how the algorithm separates the foreground image. The application in [3] is thus helpful in maritime surveillance system.
- **Animal detection:** The aim is to detect and observe animal activities in certain area. In [10] the application tries to detect birds in order to protect the aviation safety. Besides, this application can also be used to study the migration of birds and identify bird species.
- **Medical analysis:** The application in [11] gives the idea to extract the blood vessels from OCT images. The low-rank component represents the static tissues of human skin and the sparse component represents the blood vessels. By finding and analyzing a specific blood vessel, one common skin cancer can be diagnosed.

1-2-2 General applications of RPCA

Besides the background and foreground decomposition applications, RPCA is also applicable in other fields. For example, the algorithm in [12] tries to restore a video sequence from degraded one. This application is useful in restoring archived films or video that has random-valued noise. The algorithm in [2] tries to recover Optical Coherence Tomography (OCT) image which is high resolution medical image for diagnosis of disease as shown in Figure 1-2. The matrix S is in this case the speckles in the image. The application in [13] uses sparse term to classify and identify face images because sparse term captures the most discriminating feature among face images. And in [13] the algorithm models the leading vocal as sparse component of RPCA and separates it from background music.

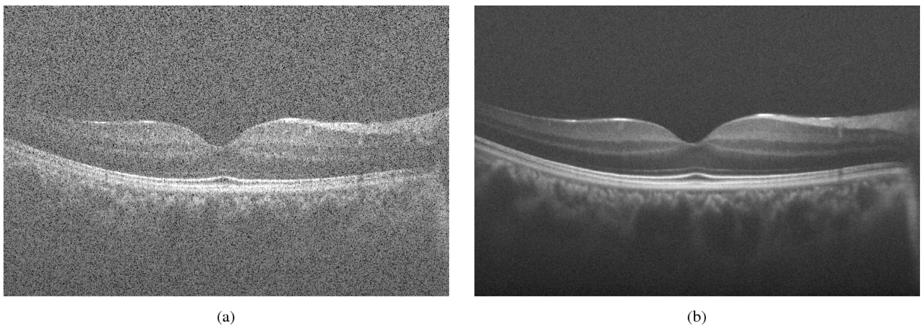

Figure 1-2: OCT image with speckle (left) and without speckle (right) [2]

1-3 Existence of negative pixels

From the applications above, RPCA is useful both for foreground extraction and image processing in other fields. However, as mentioned before, RPCA does not restrict the pixel values to be nonnegative. For example, in Figure 1-3, the foreground object boat is clearly separated from background environment. It is visible in image form because the pixels that constitute the foreground object are in the range of 0 to 255 since pixel is stored as an 8-bit integer that has this specific range. 0 is taken as black and 255 is taken as white [14]. However, the decomposition process cannot guarantee all the pixels are in this specific range. Suppose the foreground object is composed of negative pixels, it is thus not visible in image form though it is separated successfully as sparse component.

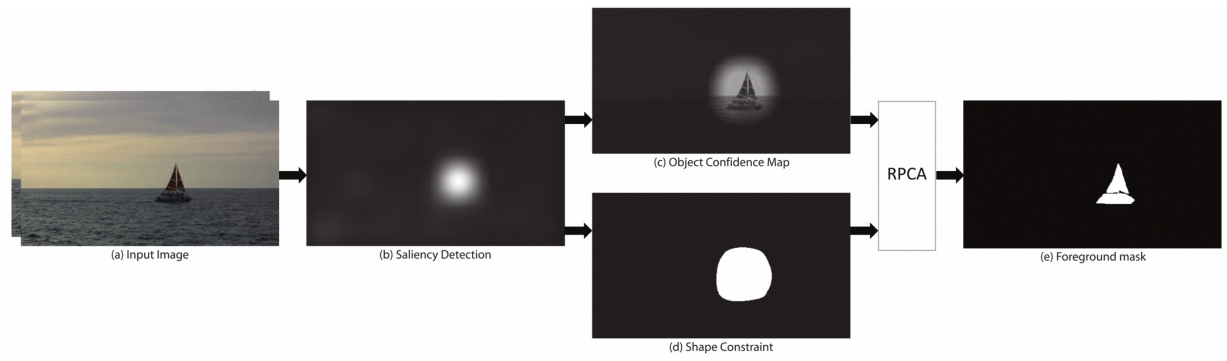

Figure 1-3: Diagram of foreground object separation [3]

Then an experiment with the help Inexact ALM (IALM) algorithm which is a common algorithm of RPCA will be given to further demonstrate the effect of negative pixels in this section. IALM is a common algorithm to solve RPCA problem. The dataset is chosen from CIFAR-100 dataset [15]. It consists of 32×32 tiny colour images in 100 classes with 600 images in each class where 50000 are training images and the rest test images. The classes include animals, people, household furniture, vehicles, etc. Before implementing algorithm,the test images were taken out and preprocessed to matrix form since they were stored as long vectors. And the RGB images were changed to grayscale images using the command **rgb2gray**.

With the help of IALM those images were successfully decomposed into low-rank and sparse components. However, it was found that 535 out of 10000 images have negative low-rank and sparse components which demonstrates that negative pixels exist even though these images were decomposed successfully. For instance, one image from the dataset is taken out to show the effect of negative pixels. Since negative pixels are meaningless, they are first replaced by zeros instead. And if all the negative pixels in S are set to zero, the result will lose much information of the foreground object as shown in Figure 1-4.

Figure 1-4: Sunflower (left) and its sparse component (right)

The sparse component becomes invisible due to the negative pixels that RPCA brings. Based on the experiment above it verifies the existence of negative pixels in RPCA algorithm. And these negative pixels can be a problem since they cannot be visible in images. Thus some important features of foreground objects will be lost. To deal with these negative pixels, in the next section one approach will be introduced to decompose the image data in a nonnegative way.

1-4 Introduction to NMF

NMF became widely known in the paper of Lee and Seung [16]. Similar to RPCA, it is also an approach that approximate high-dimensional data by factorizing the data matrix into a product of two matrices as in Eq. (1-2)

$$D = UV, \tag{1-2}$$

where $U \in R^{m\times k}$ and $V \in R^{k\times n}$. Matrix U is basis matrix and matrix V is encoding matrix. The encoding matrix V contains the coefficients that can approximate each column of D by using basis matrix. Different from RPCA, U and V which are decomposed from D are both nonnegative. The nonnegative constraint leads to a parts-based representation of the original data [17]. It means each column of the original data matrix is represented by a linear combination of basis matrix and the constraint only allows additive combinations. NMF is widely used in computer vision, document clustering and statistical classification. According to [18], there are different types of NMF algorithms besides the standard NMF. For example, Constrained NMF (CNMF) imposes additional constraints such as sparsity and orthogonality on the decomposed data matrix. Structured NMF (SNMF) directly modifies the regular factorization formula such as adding a weight matrix. In the next section the general applications of NMF will be introduced.

1-5 Applications of NMF

Due to the advantages of parts-based representation of the images, NMF is widely used in image classification and recognition. In [19], NMF extracts the feature vectors in a set of face images and it maximizes the between-class difference by adding additional constraint on standard NMF. NMF is also used to recover the measurements in reflectance spectroscopy in [20]. The spectral data provides information about light that has been reflected or scattered from a solid, liquid or gas, which can be used to identify the chemical composition of a material by examining the spectrum of the radiation from the material elements. The spectral data can also be used to identify space objects as [21] suggests. The information in the spectrum contains space objects like satellites, rocket bodies, debris and asteroids that are obtained from astronomical spectrometer. These objects are composed of different material thus the spectrum data of them differ from each other. NMF can also extract and cluster useful information from messages and mails like the application in [17]. The algorithm in [17] tries to find the main topics together with its corresponding key words that describe those topics. These key words are selected because they have the largest magnitude for each corresponding feature vector, which means they can best represent those selected topics.

1-6 Objective and structure of the thesis

As introduced in section 1-4, NMF can not only represent the original high-dimensional data matrix but adds the nonnegative constraint to the decomposed data as well. It provides an approach to force the negative image pixels to be nonnegative. Then it is necessary to examine the feasibility of incorporating the nonnegative feature of NMF into the structure of RPCA. The objective of the thesis is to find a feasible algorithm that combines RPCA and NMF with its application in background and foreground image decomposition.

In chapter 1 the applications of RPCA and NMF have already been introduced. In chapter 2, the standard algorithms of RPCA and NMF will be explained first. Based on that the proposed algorithms will be introduced in detail. The main difference of the proposed algorithm with RPCA lies in the fact that it forces the negative pixels in sparse component to be nonnegative with the help of NMF. It will mainly explain the update rules and operators that are used to solve the optimal solution. In chapter 3, the experiment of proposed algorithms together with RPCA algorithm will be made to decompose the chosen image dataset. It will evaluate the quality of the decomposed images and then compare the two algorithms with RPCA algorithm in a quantitative way. Finally in chapter 4 the advantages and disadvantages of the proposed algorithm will be analysed over various aspects and possible improvements of the algorithm will be raised.

Chapter 2

Algorithm

In this chapter, basic models of RPCA and NMF will first be introduced. On the basis of these two models, two newly proposed models will then be explained in detail which includes their model, cost function and update rules. State-of-the-art algorithms that have been used in foreground extraction will be briefly introduced in section 2-5. In the last section of this chapter RPCA and NMF algorithms that are adopted in literature will be briefly introduced and compared.

2-1 RPCA model

Assume $D \in R^{m \times n}$ is the observed data matrix. $L \in R^{m \times n}$ is the low-rank matrix and $S \in R^{m \times n}$ is the sparse matrix. The basic RPCA model can be expressed as:

$$D = L + S. \tag{2-1}$$

Following the definition of L and S, there are two things in Eq. (2-1) that need to be optimized: the rank of the L and the sparsity of S. The aim is to find L that gives the lowest rank and S that is sparse. The optimization problem can be expressed as

$$\min_{L,S} \operatorname{rank}(L) + \lambda \|S\|_0, \tag{2-2}$$

where $\| \cdot \|_0$ represents l_0-norm that is the number of nonzero elements in the matrix. The scalar λ, which is a nonnegative real number, is a constant that acts as a trade-off between low-rank and sparse component.
However, Eq. (2-2) is highly non-convex and non-smooth which means it is difficult to solve. In [9] the model uses a convex relaxation of the original optimization problem. This tractable optimization problem replaces the rank of L with nuclear norm which is the sum of singular values of a matrix. The convex relaxation also replaces l_0-norm with l_1-norm which is the maximum absolute column sum of a matrix. The convex optimization problem becomes:

$$\begin{gathered} \min_{L,S} \|L\|_* + \lambda \|S\|_1, \\ \text{s.t. } D = L + S. \end{gathered} \tag{2-3}$$

According to Karush–Kuhn–Tucker (KKT) conditions [22], the equality constraint of the optimization problem can be incorporated into the problem with the help of Lagrange multiplier. The cost function of the optimization problem can then be written as:

$$\mathcal{L}(L,S,Y) = \|L\|_* + \lambda\|S\|_1 + \langle Y, D-L-S\rangle + \frac{\mu}{2}\|D-L-S\|_F^2. \tag{2-4}$$

$Y \in R^{m\times n}$ is the Lagrange multiplier and $\langle\cdot\rangle$ represents matrix inner product that can also be written in the trace form of matrix product as $\mathrm{tr}(Y(D-L-S)^T)$. The last term calculates the error between recovered L and S with the data matrix D. The scalar μ, which is a nonnegative real number, is a constant that balances the error term.
The cost function can also be written with the error term only as:

$$\mathcal{L}(L,S,Y) = \|L\|_* + \lambda\|S\|_1 + \frac{\mu}{2}\|D-L-S\|_F^2, \tag{2-5}$$

which is enough to incorporate the equality constraint into the cost function. However, to minimize the cost function, the scalar μ needs to be set extremely large in order to give small error between D and $L+S$. And with the help of Lagrange multiplier Y which will be updated in each iteration, it is not necessary to set μ as large as possible to minimize the error. And by using Y, the optimization problem will converge to the optimal faster because Y is improved at every iteration.

2-2 NMF model

Assume $D \in R^{m\times n}$ is the data matrix. NMF approximates D matrix by using a product of two nonnegative matrices $U \in R^{m\times k}$ and $V \in R^{k\times n}$ as $D \approx UV$. Each column of U is called basis vector while each column of V is called encoding vector corresponding with each column of D. The product of U and V can be regarded as a compressed expression of the original data matrix D. Each column of D is approximated by a linear combination of U weighted by each column of V [23]. Scalar k is the rank of matrices U and V. In [16] the rank is chosen to be $k < \frac{mn}{m+n}$. The rank choice is made based on the quality of approximation. If it is too small the approximation will lose important features in D. If it is too large the dimension reduction is not achieved. The optimization problem is to minimize the difference between the recovered matrices and the original data matrix with the nonnegative constraint. It is defined as:

$$\begin{gathered}\mathcal{L}(U,V) = \|D-UV\|_F^2,\\ \text{s.t. } U \geq 0, V \geq 0,\end{gathered} \tag{2-6}$$

where $\|\cdot\|_F$ is the Frobenius norm which is defined as $\|D\|_F = \sqrt{\sum_i^m \sum_j^n |d_{ij}|^2} = \sqrt{\sum_{i=1}^{\min(m,n)} \sigma_i^2(D)}$ [24]. The scalar σ_i represents the i-th singular value of matrix D. The nonnegative constraint on U and V means every element in matrix U and V are nonnegative.

2-2-1 Validation test on NMF

A validation test was made on standard NMF algorithm to verify if these two components U and V are all nonnegative. The same dataset CIFAR-100 is used as in section 1-3. The grayscale image is still in the form of 32×32 square matrix. The stopping criteria is defined as the relative improvement of two successive iterations as $\frac{\|\hat{D}(k+1)-\hat{D}(k)\|_F^2}{\|\hat{D}(k+1)\|_F^2} \leq \epsilon$ where ϵ is a small value such as 10^{-6} and k is the iteration number. The maximum iteration number for one image was chosen to be 20000 after several trials. The nonnegativity of the two components U and V were checked for every image in the 10000 test images. It turned out that there were no negative entries in these two components U and V for every image in this test.

2-3 Proposed model 1

2-3-1 Main optimization problem

In chapter 1 it is known that the pixel in the recovered result from RPCA can be negative, which exists both in low-rank and sparse component. One possible solution is to force the pixel to be nonnegative by adding the nonnegative constraint using the model of NMF. In section 2-2-1 the test verifies the nonnegativity of the decomposed components. Suppose there are negative pixels in sparse component, it is feasible to decompose sparse component as

$$\begin{aligned} S &= U_S V_S, \\ \text{s.t. } U_S &\geq 0, V_S \geq 0. \end{aligned} \tag{2-7}$$

By adding these two constraints, the optimization problem becomes

$$\begin{gathered} \|L\|_* + \lambda \|S\|_1, \\ \text{s.t. } D = L + S, \\ S = U_S V_S, \\ U_S \geq 0,\ V_S \geq 0. \end{gathered} \tag{2-8}$$

where $D \in R^{m\times n}$, $L \in R^{m\times n}$ and $S \in R^{m\times n}$. $U_S \in R^{m\times k}$ and $V_S \in R^{k\times n}$ are the nonnegative decomposition of sparse component. By using Lagrange multiplier and error function as in RPCA model, the corresponding cost function is defined as in Eq. (2-9),

$$\begin{aligned} \mathcal{L}(L,S,U_S,V_S) = \|L\|_* + \lambda \|S\|_1 + \langle Y_D, D-L-S\rangle + \tfrac{\mu}{2}\|D-L-S\|_F^2 \\ + \langle Y_S, S - U_S V_S\rangle + \tfrac{\mu}{2}\|S-U_SV_S\|_F^2 + \tfrac{\mu}{2}\|U_S - C_1\|_F^2 + \tfrac{\mu}{2}\|V_S - C_2\|_F^2. \end{aligned} \tag{2-9}$$

Matrices $C_1 \in R^{m\times k}$ and $C_2 \in R^{k\times n}$ are both zero matrix since all the elements in matrices U_S and V_S are greater than zero in the inequality constraint. The last two error functions are penalty functions that incorporate the constraint of U_S and V_S into the cost function. To simplify the cost function and update rules, the same μ is used as regularization term for both equality and inequality constraint. The reason for using the same μ is that μ is not updated throughout the whole process. And with the update of Lagrange multipliers the error between D and $L+S$ will gradually be minimized thus it is not necessary to set different μ for every error term.
Next the problem is to find the optimal solution L^*, S^*, U_S^* and V_S^*. It is feasible to divide the main optimization problem into four sub-problems with respect to these four components respectively. The sub-problem to solve low-rank component is

$$\mathcal{L}_L = \|L\|_* + \langle Y_D, D-L-S\rangle + \frac{\mu}{2}\|D-L-S\|_F^2. \tag{2-10}$$

The sub-problem to solve sparse component is

$$\mathcal{L}_S = \lambda \|S\|_1 + \langle Y_D, D-L-S\rangle + \frac{\mu}{2}\|D-L-S\|_F^2 + \langle Y_S, S-U_SV_S\rangle + \frac{\mu}{2}\|S-U_SV_S\|_F^2\,. \tag{2-11}$$

The sub-problem to solve U_S term is

$$\mathcal{L}_{U_S} = \langle Y_S, S-U_SV_S\rangle + \frac{\mu}{2}\|S-U_SV_S\|_F^2 + \frac{\mu}{2}\|U_S-C_1\|_F^2\,. \tag{2-12}$$

The sub-problem to solve V_S term is

$$\mathcal{L}_{V_S} = \langle Y_S, S-U_SV_S\rangle + \frac{\mu}{2}\|S-U_SV_S\|_F^2 + \frac{\mu}{2}\|V_S-C_2\|_F^2\,. \tag{2-13}$$

2-3-2 Sub-problem of L

To solve the optimal of L, it is first separated from other terms in Eq. (2-10) as

$$\begin{aligned}\mathcal{L}_L &= \mu^{-1}\|L\|_* + \tfrac{1}{2}\left\|L-(D-S+\mu^{-1}Y_D)\right\|_F^2\\ &= \mu^{-1}\|L\|_* + \tfrac{1}{2}\|L\|_F^2 - \langle L, D-S+\mu^{-1}Y_D\rangle + \tfrac{1}{2}\left\|D-S+\mu^{-1}Y_D\right\|_F^2\,.\end{aligned} \tag{2-14}$$

According to [25], the inner product of two matrices is upper bounded by the product of their singular values, which means the cost function has its minimum when the inner product reaches the upper bound. Then Eq. (2-14) can be expressed in singular value form as

$$\mathcal{L}_L = \mu^{-1}\sum_{i=1}^{\min(m,n)}\sigma_i(L) + \tfrac{1}{2}\sum_{i=1}^{\min(m,n)}\sigma_i(L)^2 - \sum_{i=1}^{\min(m,n)}\sigma_i(L)\sigma_i(P) + \tfrac{1}{2}\sum_{i=1}^{\min(m,n)}\sigma_i(P)^2, \tag{2-15}$$

where $\sigma(L)$ is the singular value of L and $\sigma(P)$ is the singular value of $D-S+\mu^{-1}Y_D$. Scalars m and n are the matrix dimension.
The optimal solution of singular value of L can be obtained by differentiating Eq. (2-15) with respect to the singular value of L and setting it equal to zero,

$$\begin{aligned}\frac{\partial\mathcal{L}_L}{\partial\sigma_i(L)} = \sum_{i=1}^{\min(m,n)}\mu^{-1} + \sum_{i=1}^{\min(m,n)}\sigma_i(L) - \sum_{i=1}^{\min(m,n)}\sigma_i(P) = 0,\\ \sigma_i(L) = \sigma_i(P) - \mu^{-1}.\end{aligned} \tag{2-16}$$

It is found that only when $\sigma_i(L) = \sigma_i(P) - \mu^{-1}$, the optimal solution of L can be obtained. In [9] this calculation process is achieved with the help of singular value thresholding operator $\mathcal{D}$ that is defined as

$$\begin{aligned}(U,\Sigma,V) = \text{svd}(D-S+\mu^{-1}Y_D),\\ \mathcal{S}_\tau(\Sigma) = \text{sgn}(\Sigma)\text{max}(|\Sigma|-\tau,0),\\ L^* = \mathcal{D}_\tau(D-S+\mu^{-1}Y_D) = U\mathcal{S}_\tau(\Sigma)V^T.\end{aligned} \tag{2-17}$$

The operator $\mathcal{D}$ first decomposes $D-S+\mu^{-1}Y_D$ to find its singular values by using Singular Value Decomposition (SVD) and adds the threshold to every singular value of $D-S+\mu^{-1}Y_D$ where $\tau = \mu^{-1}$ in Eq. (2-17). After obtaining the singular values of L, they are multiplied by the left and right singular vectors again to get the optimal solution L^*. The max-operator here compares every diagonal element of the singular value matrix with zero and replace all the negative values with zero. Those negative values represent that the singular values are below the threshold.

2-3-3 Sub-problem of S

Similar to sub-problem of L, the term S is firstly separated from other terms in Eq. (2-11),

$$\mathcal{L}_S = \lambda\mu^{-1}\left\|S\right\|_1 + \tfrac{1}{2}\left\|S - (D - L + \mu^{-1}Y_D)\right\|_F^2 + \tfrac{1}{2}\left\|S - (U_S V_S - \mu^{-1}Y_S)\right\|_F^2. \tag{2-18}$$

Then the optimal solution of S can be obtained by differentiating Eq. (2-18) with respect to S and setting it equal to zero,

$$\begin{aligned} \tfrac{\partial\mathcal{L}_S}{\partial S} &= \lambda\mu^{-1}\tfrac{\partial\|S\|_1}{\partial S} + (S - (D - L + \mu^{-1}Y_D) + (S - (U_S V_S - \mu^{-1}Y_S)) = 0, \\ S &= \tfrac{1}{2}(D - L + \mu^{-1}Y_D + U_S V_S - \mu^{-1}Y_S - \lambda\mu^{-1}\tfrac{\partial\|S\|_1}{\partial S}). \end{aligned} \tag{2-19}$$

Since it is assumed that S is nonnegative, $\|S\|_1$ can be simplified as S itself. The derivative of S with respect to S itself is 1. Eq. (2-19) is then simplified as

$$\begin{aligned} S &= \tfrac{1}{2}(D - L + \mu^{-1}Y_D + U_S V_S - \mu^{-1}Y_S - \lambda\mu^{-1}) \\ &= \tfrac{1}{2}(D - L + \mu^{-1}Y_D + U_S V_S - \mu^{-1}Y_S) - \tfrac{\lambda}{2\mu} \end{aligned} \tag{2-20}$$

With the help of shrinkage operator again the optimal solution S^* can be represented as

$$S^* = \mathcal{S}_{\frac{\lambda}{2\mu}}\frac{1}{2}(D - L + \mu^{-1}Y_D + U_S V_S - \mu^{-1}Y_S), \tag{2-21}$$

where the threshold is $\frac{\lambda}{2\mu}$ in this case.

2-3-4 Sub-problems of U_S and V_S

For the optimal solution of U_S and V_S, it is feasible to simply take the derivative of the sub-problems with respect to U_S and V_S and set them equal to zero since there is no l_1-norm or nuclear norm in their cost function. The optimum U_S^* is,

$$\begin{aligned} \mathcal{L}_{U_S} &= \tfrac{\mu}{2}\left\|S + \mu^{-1}Y_S - U_S V_S\right\|_F^2 + \tfrac{\mu}{2}\left\|U_S - C_1\right\|_F^2, \\ \tfrac{\partial\mathcal{L}_{U_S}}{\partial U_S} &= \mu(S + \mu^{-1}Y_S - U_S V_S)(-V_S) + \mu U_S - \mu C_1 \\ &= -\mu S V_S - Y_S V_S + \mu U_S V_S^2 + \mu U_S - \mu C_1 = 0, \\ U_S^* &= (\mu S {V_S}^T + Y_S {V_S}^T + \mu C_1)(\mu(I + V_S {V_S}^T))^{-1}. \end{aligned} \tag{2-22}$$

By using the same strategy, the optimal solution of V_S^* can be obtained as $V_S^* = (\mu(I + {U_S}^T U_S))^{-1}(\mu {U_S}^T S + {U_S}^T Y_S + \mu C_2)$.

2-3-5 Update rules

The optimal solutions for the low-rank component, sparse component, U_S term and V_S term are

$$\begin{aligned} L^* &= \mathcal{D}_{\frac{1}{\mu}}(D - S + \mu^{-1}Y_D), \\ S^* &= \mathcal{S}_{\frac{\lambda}{2\mu}}(\tfrac{1}{2}(D - L + \mu^{-1}Y_D + U_S V_S - \mu^{-1}Y_S)), \\ U_S^* &= \max((\mu S {V_S}^T + Y_S {V_S}^T + \mu C_1)(\mu(I + V_S {V_S}^T))^{-1}, 0), \\ V_S^* &= \max((\mu(I + {U_S}^T U_S))^{-1}(\mu {U_S}^T S + {U_S}^T Y_S + \mu C_2), 0). \end{aligned} \tag{2-23}$$

All of these components are updated in each iteration with the calculated optimal solutions. After all the components are updated, the Lagrange multipliers are then updated as well,

$$\begin{aligned} Y_D &= Y_D + \mu(D - L - S), \\ Y_S &= Y_S + \mu(S - U_S V_S). \end{aligned} \tag{2-24}$$

At this point, the update of the first iteration ends. The updated values will be kept as the initial values of the next iteration. The whole process continues until the stopping criterion is fulfilled. Stopping criterion is defined as the relative error between the recovered data matrix and the original data matrix as

$$\begin{aligned} \text{error}_D &= \frac{\|D-L-S\|_F}{\|D\|_F} \le \rho, \\ \text{error}_S &= \frac{\|S-U_S V_S\|_F}{\|S\|_F} \le \rho, \end{aligned} \tag{2-25}$$

where ρ is a relatively small value such as 10^{-6}.
The whole algorithm is generalized in algorithm 1.

Algorithm 1 Nonnegative Sparse Component Robust PCA

```
Input: $D \in R^{m\times n}$, $\lambda > 0$, $\mu > 0$, $L_0 \in R^{m\times n}$, $S_0 \in R^{m\times n}$, $k > 0$.
Output: $L^*$, $S^*$, $U_S^*$, $V_S^*$.
1: while not converged do
2:     $L = \mathcal{D}_{\frac{1}{\mu}}(D - S_k + \mu^{-1} Y_D)$;
3:     $S = \mathcal{S}_{\frac{\lambda}{2\mu}}(\frac{1}{2}(D - L_{k+1} + \mu^{-1} Y_D + U_S V_S - \mu^{-1} Y_S))$;
4:     $U_S = \max((\mu S {V_S}^T + Y_S {V_S}^T + \mu C_1)(\mu(I + V_S {V_S}^T))^{-1}, 0)$;
5:     $V_S = \max((\mu(I + {U_S}^T U_S))^{-1}(\mu {U_S}^T S + {U_S}^T Y_S + \mu C_2), 0)$;
6:     $Y_D = Y_D + \mu(D - L - S)$;
7:     $Y_S = Y_S + \mu(S - U_S V_S)$;
8: end while
```

Similarly if the goal is to set the low-rank component to be nonnegative, the optimization problem becomes

$$\begin{gathered} \|L\|_* + \lambda \|S\|_1, \\ \text{s.t. } D = L + S, \\ L = U_L V_L, \\ U_L \ge 0,\ V_L \ge 0. \end{gathered} \tag{2-26}$$

By incorporating the constraints into the model, the corresponding cost function is defined as

$$\begin{aligned} \mathcal{L}(L, S, U_L, V_L) = \|L\|_* + \lambda \|S\|_1 + \langle Y_D, D - L - S\rangle + \tfrac{\mu}{2}\|D - L - S\|_F^2 \\ + \langle Y_L, L - U_L V_L\rangle + \tfrac{\mu}{2}\|L - U_L V_L\|_F^2 + \tfrac{\mu}{2}\|U_L - C_1\|_F^2 + \tfrac{\mu}{2}\|V_L - C_2\|_F^2. \end{aligned} \tag{2-27}$$

The update rules is obtained by grouping all the terms that have L and S separately like the strategy in section 2-3-1 as

$$\mathcal{L}_L = \|L\|_* + \langle Y_D, D - L - S\rangle + \frac{\mu}{2}\|D - L - S\|_F^2 + \langle Y_L, L - U_L V_L\rangle + \frac{\mu}{2}\|L - U_L V_L\|_F^2, \tag{2-28}$$

$$\mathcal{L}_S = \lambda\|S\|_1 + \langle Y_D, D - L - S\rangle + \frac{\mu}{2}\|D - L - S\|_F^2, \tag{2-29}$$

$$\mathcal{L}_{U_L} = \langle Y_L, L - U_L V_L\rangle + \frac{\mu}{2}\left\|L - U_L V_L\right\|_F^2 + \frac{\mu}{2}\left\|U_L - C_1\right\|_F^2, \tag{2-30}$$

$$\mathcal{L}_{V_L} = \langle Y_L, L - U_L V_L\rangle + \frac{\mu}{2}\left\|L - U_L V_L\right\|_F^2 + \frac{\mu}{2}\left\|V_L - C_2\right\|_F^2. \tag{2-31}$$

Then use the same method to solve the sub-problems of L, S, U_L and V_L to find their optimal solutions. The update rules for the low-rank component to be nonngative can be expressed as,

$$\begin{aligned} L^* &= \mathcal{D}_{\frac{1}{2\mu}}(D - S + \mu^{-1}Y_D + U_L V_L - \mu^{-1}Y_L), \\ S^* &= \mathcal{S}_{\frac{\lambda}{\mu}}(D + \mu^{-1}Y_D - L), \\ U_L^* &= \max((\mu S {V_S}^T + Y_S {V_S}^T + \mu C_1)(\mu(I + V_S {V_S}^T))^{-1}, 0), \\ V_L^* &= \max((\mu(I + {U_L}^T U_L))^{-1}(\mu {U_L}^T L + {U_L}^T Y_L + \mu C_2), 0). \end{aligned} \tag{2-32}$$

The whole algorithm is generalized in algorithm L1.

Algorithm L1 Nonnegative Low-rank Component Robust PCA

Input: $D \in R^{m\times n}$, $\lambda > 0$, $\mu > 0$, $L_0 \in R^{m\times n}$, $S_0 \in R^{m\times n}$, $k > 0$.
Output: L^*, S^*, U_L^*, V_L^*.

```
1: while not converged do
2:     $L = \mathcal{D}_{\frac{1}{2\mu}}(D - S + \mu^{-1}Y_D + U_L V_L - \mu^{-1}Y_L)$;
3:     $S = \mathcal{S}_{\frac{\lambda}{\mu}}(D + \mu^{-1}Y_D - L)$;
4:     $U_L = \max((\mu S {V_S}^T + Y_S {V_S}^T + \mu C_1)(\mu(I + V_S {V_S}^T))^{-1}, 0)$;
5:     $V_L = \max((\mu(I + {U_L}^T U_L))^{-1}(\mu {U_L}^T L + {U_L}^T Y_L + \mu C_2), 0)$;
6:     $Y_D = Y_D + \mu(D - L - S)$;
7:     $Y_L = Y_L + \mu(L - U_L V_L)$;
8: end while
```

In algorithm 1 and L1, the input parameters need to be chosen beforehand. As suggested in [9], [26] and [27], the value of λ is generally chosen to be $\frac{1}{\sqrt{max(m,n)}}$ and μ is $\frac{mn}{4\|D\|_1}$. As revealed in [9], it is not very clear why the choice of λ is correct no matter what the initialization of L and S are. But the recovery error error_D and error_S keep decreasing in the experiment process, which means the error between the recovered data and original data minimizes if the parameters are chosen this way. It gives a hint to choose the initial parameters around the given values.

2-4 Proposed model 2

2-4-1 Main optimization problem

In proposed model 1, the nonnegativity of S is achieved by decomposing S into U_S and V_S following the update rules of NMF. Matrices U_S and V_S are nonnegative which also represents the nonnegativity of S due to the constraint $S = U_S V_S$. But this constraint is still an approximation of S. Rather than incorporating the constraint $S = U_S V_S$ as in algorithm 1, it is more direct to replace all S with $U_S V_S$ in RPCA model. The motivation is to further simplify the cost function and update rules. By substituting all the S in the original model, the new model becomes

$$\begin{aligned} &\|L\|_* + \lambda \|U_S V_S\|_1, \\ &\text{s.t. } D = L + U_S V_S, \\ &U_S \geq 0,\ V_S \geq 0, \end{aligned} \tag{2-33}$$

where $D \in R^{m\times n}$ and $L \in R^{m\times n}$. $U_S \in R^{m\times k}$ and $V_S \in R^{k\times n}$ are the nonnegative decomposition of sparse component. Its cost function is obtained by also using Lagrange multiplier as in 2-3-1. The corresponding cost function is defined as in Eq. (2-34),

$$\begin{aligned} \mathcal{L}(L, U_S, V_S) = \|L\|_* + \lambda \|U_S V_S\|_1 + \langle Y_D, D - L - U_S V_S\rangle + \tfrac{\mu}{2} \|D - L - U_S V_S\|_F^2 \\ + \tfrac{\mu}{2} \|U_S - C_1\|_F^2 + \tfrac{\mu}{2} \|V_S - C_2\|_F^2 . \end{aligned} \tag{2-34}$$

Next the problem is to find the optimal solution L^*, U_S^* and V_S^*. The main optimization problem is divided into three sub-problems with respect to these three components respectively. The sub-problem to solve low-rank component is

$$\mathcal{L}_L = \|L\|_* + \langle Y_D, D - L - U_S V_S\rangle + \frac{\mu}{2} \|D - L - U_S V_S\|_F^2 . \tag{2-35}$$

The sub-problem to solve U_S term is

$$\mathcal{L}_{U_S} = \lambda \|U_S V_S\|_1 + \langle Y_D, D - L - U_S V_S\rangle + \frac{\mu}{2} \|D - L - U_S V_S\|_F^2 + \frac{\mu}{2} \|U_S - C_1\|_F^2 . \tag{2-36}$$

The sub-problem to solve V_S term is

$$\mathcal{L}_{V_S} = \lambda \|U_S V_S\|_1 + \langle Y_D, D - L - U_S V_S\rangle + \frac{\mu}{2} \|D - L - U_S V_S\|_F^2 + \frac{\mu}{2} \|V_S - C_2\|_F^2 . \tag{2-37}$$

2-4-2 Update rules

Since the cost function for L is the same as in Eq. (2-10). The optimum L^* is obtained when $\sigma_i(L) = \sigma_i(D + \mu^{-1} Y_D - U_S V_S) - \mu^{-1}$. Here $\sigma_i(L)$ is the singular value of L while $\sigma_i(D + \mu^{-1} Y_D - U_S V_S)$ is the singular value of $D + \mu^{-1} Y_D - U_S V_S$. The optimum L^* is:

$$\begin{aligned} (U, \Sigma, V) &= \text{svd}(D + \mu^{-1} Y_D - U_S V_S), \\ \mathcal{S}_\tau(\Sigma) &= \text{sgn}(\Sigma)\text{max}(|\Sigma| - \tau, 0), \\ L^* = \mathcal{D}_\tau(D + \mu^{-1} Y_D - U_S V_S) &= U \mathcal{S}_\tau(\Sigma) V^T . \end{aligned} \tag{2-38}$$

For the updates of U_S and V_S, by taking the derivative of their corresponding cost function and setting it equal to zero, the optimum U_S^* and V_S^* can be obtained,

$$\begin{aligned} U_S^* &= \max((D{V_S}^T + \mu^{-1} Y_D {V_S}^T - L V_S^T + C_1 - \text{sign}(U_S V_S) V_S^T \lambda \mu^{-1})(I + V_S {V_S}^T)^{-1}, 0), \\ V_S^* &= \max((I + {U_S}^T U_S)^{-1}({U_S}^T D + \mu^{-1} {U_S}^T Y_D - U_S^T L + C_2 - U_S^T \text{sign}(U_S V_S) \lambda \mu^{-1}), 0). \end{aligned} \tag{2-39}$$

The whole algorithm is summarized in algorithm 2.

Algorithm 2 Nonnegative Sparse Component Robust PCA 2

```
Input: $D \in R^{m\times n}$, $\lambda > 0$, $\mu > 0$, $L_0 \in R^{m\times n}$, $U_{S0} \in R^{m\times k}$, $V_{S0} \in R^{k\times n}$, $k > 0$.
Output: $L^*$, $S^*$.
1: while not converged do
2:     $L = \mathcal{D}_{\frac{1}{\mu}}(D - S_k + \mu^{-1}Y_D)$;
3:     $U_S = \max((DV_S{}^T + \mu^{-1}Y_D V_S{}^T - LV_S^T + C_1 - \mathrm{sign}(U_S V_S)V_S^T\lambda\mu^{-1})(I + V_S V_S{}^T)^{-1}, 0)$;
4:     $V_S = \max((I + U_S{}^T U_S)^{-1}(U_S{}^T D + \mu^{-1}U_S{}^T Y_D - U_S^T L + C_2 - U_S^T\mathrm{sign}(U_S V_S)\lambda\mu^{-1}), 0)$;
5:     $Y_D = Y_D + \mu(D - L - S)$;
6:     $S = U_S V_S$;
7: end while
```

2-5 Connection with other algorithms for background and foreground decomposition

In this section the connection of the proposed algorithm with other state-of-the-art algorithms used in background and foreground image decomposition will be discussed, such as the algorithms used in the applications in chapter 1. This section will mainly discuss their similarities or differences in terms of aspects like update rules and complexity.

- **Augmented Lagrange Multiplier Method (ALM):** The application in [6] uses Inexact ALM (IALM) to solve its optimization problem. IALM is originally from algorithm Exact ALM (EALM). For EALM and IALM, they both use Lagrange multiplier to incorporate the equality constraint into its cost function as

$$\mathcal{L}(L, S, Y) = \|L\|_* + \lambda\|S\|_1 + \langle Y, D - L - S\rangle + \frac{\mu}{2}\|D - L - S\|_F^2. \tag{2-40}$$

  The difference between these two algorithms is that the update number of L and S is different. They both will solve the optimal solution in the update rules. For EALM, the update continues until exact optimal solution of L and S is obtained in the sub-problem. For IALM, L and S only update once in the sub-problem but it is sufficient for L and S to converge to the optimal solution of the main RPCA optimization problem. Due to different number of updates, the complexity to solve the the optimization per iteration is different as well. For EALM the computation load in each iteration is dominated by SVD which has complexity $\mathcal{O}$(mn min(m,n)) [26]. While for IALM since there is only one update for the sub-problem of L and S in each iteration, the use of SVD is then less than that of EALM. The complexity of it is $\mathcal{O}$(rmn) where r is the rank of L. For the proposed algorithm, there is only one update for L and S in each iteration like IALM algorithm. The complexity per iteration is the same as that of IALM.

- **Alternating Direction Method (ADM):** In the application of [3], the optimization problem is formed with the help of Lagrange multiplier in ALM. But the update process uses ADM for the update of L and S respectively. In [26] ADM is described as an improvement of the classical ALM. The difference between ADM and ALM lies in the fact that L and S are updated simultaneously in ALM while serially in ADM. In RPCA problem, the algorithm will first minimize the cost function with respect to L. With this newly updated L the algorithm minimizes the cost function with respect to S. When the optimal solutions of L and S are obtained, the algorithm then updates Lagrange multiplier. In the proposed algorithm, this update scheme is also adopted.

- **Fast PCP (FPCP):** The algorithm in [28] estimates the rank of L in a different way than in ALM. In the update of L the algorithm will gradually increase the rank of L by evaluating the contribution of each singular value. If the contribution of specific singular value is within certain threshold then the rank will stop increasing. And FPCP is found to be faster than IALM in terms of computational performance in [28]. In the proposed algorithm, the rank of L is automatically generated by the algorithm. Only when L is further decomposed into U_L and V_L the rank of L can be chosen beforehand.
- **Total Variation Regularized RPCA (TVRPCA):** In the applcation of [29] the update strategy is the same as the proposed algorithm which combines ALM and ADM scheme to update L and S. But TVRPCA is also used under dynamic background condition where part of the background can be erroneously detected as foreground object. Thus TVRPCA introduces a total variation term which compares the pixel with its neighboring pixels because the foreground object is spatially continuous in the image, which means the intensity of pixels in the moving object area is continuous. TVRPCA successfully separated the moving object and dynamic background in its model. And the algorithm gives the same complexity as original RPCA in the update process [29]. TVRPCA gives a hint of how to deal with dynamic background to further improve the proposed algorithm.
- **Accelerated Proximal Gradient (APG):** The update rules of APG is different from ALM and ADM. In [30] it formulates the optimization problem as

$$\mathcal{L}(L, S, Y) = \|L\|_* + \lambda\|S\|_1 + \frac{1}{\mu}\|D - L - S\|_F^2, \tag{2-41}$$

  where there is only error term in the cost function. The update scheme does not use Lagrange multiplier. Instead, it uses the gradient of the error term to update L and S in each iteration. But the convergence rate of APG algorithm is generally slower than ALM. By varying the choice of μ in each iteration, the convergence speed can be greatly improved.

2-6 Algorithm comparison

In this section, the algorithms of RPCA and NMF that have been used in the applications mentioned in chapter 1 will be listed and their complexity to solve the optimization problem per iteration, advantages and disadvantages will be shown in Table 2-1. For RPCA algorithms, Inexact ALM has the lowest complexity per iteration because it requires less number of SVD. ALM has Q-linear convergence speed and it minimizes L and S simultaneously while ADM minimizes them serially. For NMF algorithms, Projected Gradient has the highest complexity per iteration since it will check if there is sufficient decrease for cost function. And the step size can be changed in order to converge to optimal solution faster. Constrained NMF adds penalty terms in the cost function in order to enforce certain properties such as smoothness and sparsity. For example, Fisher NMF will be used in recognition and it adds the additional term in the cost function to maximize the between-class difference and minimize with-in class difference.

Table 2-1: RPCA and NMF algorithms

Algorithm	**Complexity per iteration**	**Advantages and disadvantages**
	RPCA algorithm	
ADM [26]	$\mathcal{O}$(mn min(m,n))	It minimizes L and S serially. The computational cost is less than ALM.
ALM [2]	$\mathcal{O}$(mn min(m,n))	It has Q-liner convergence speed compared to sub-linear speed of APG. It minimizes L and S simultaneously.
Inexact ALM [31]	$\mathcal{O}$(rmn)	It requires less number of partial SVD. It is generally faster than APG and the recovery precision is higher than APG.
APG [12]	$\mathcal{O}$(mn min(m,n)) with full SVD	It is applicable to large-scale data but the convergence rate is slower than that of ALM.
	NMF algorithm	
Standard NMF [16]	$\mathcal{O}$(rmn)	It is the standard update scheme. The rank of the recovered matrix is set in advance.
Gradient Descent with Constrained Least Squares [17]	$\mathcal{O}$(rmn)	It enforces smoothness and sparsity on basis matrix U with the use of penalty terms.
Constrained NMF [21]	$\mathcal{O}$(rmn)	It enforces smoothness on basis matrix U and other constraints can also be added with the use of penalty terms.
Fisher NMF [19]	$\mathcal{O}$(rmn)	It maximizes the between-class scatter and minimize the within-class scatter of encoding matrix V. It increases the recognition rate.

Continued on next page

Table 2-1 – continued from previous page

Algorithm	**Complexity per iteration**	**Advantages and disadvantages**
Projected Gradient [32]	$\mathcal{O}$(trmn) t is the average number of sufficient decrease condition for cost function checked per iteration	The step size can be chosen and changed during iteration. It can lead to faster convergence if step size is chosen correctly.

Chapter 3

Experiment

3-1 Dataset

In this chapter, one application is developed based on the proposed algorithm in chapter 2. The algorithm will be used to decompose the data image and extract the foreground or background components. The dataset is chosen from UCSD Background Subtraction Dataset [33]. It consists of 18 video sequences with each frame of the sequence in JPEG format. The video sequence includes grayscale image sequence of natural scenery where there is moving object in the foreground. It also has the ground truth images that correspond to each frame of the sequence. In ground truth data 255 indicates foreground and 0 indicates background. The ground truth data is seen as a comparison with the sparse component which is decomposed using the proposed algorithm.
In order to simplify the dataset, each image frame is stored as a column vector in a large data matrix with dimension equals [length of one image frame $\times$ number of frames]. Such as the bottle video sequence, it has 31 frames where the length of each frame equals 68096 so the dimension of data matrix is $D \in R^{68096 \times 31}$. The foreground object of this video sequence is a drifting bottle on the sea surface. To separate the foreground, the proposed algorithm is firstly used to the data matrix D. Background and foreground matrices of all the frames can be obtained. Then background and foreground component of each frame can be taken out separately. By reshaping each frame into the dimension of original image matrix $R^{224 \times 304}$, the foreground and background components can be visible in images.

3-2 Evaluation criterion

In this section the performance measures for comparing the result of different algorithms will be introduced, including the definitions and formulas of each performance measure.

3-2-1 Mean Squared Error (MSE) and Peak Signal-to-noise Ratio (PSNR)

The difference between recovered foreground matrix S with the ground truth data S_0 is defined as MSE,

$$\text{MSE } = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[S(i,j)-S_0(i,j)]^2 = \frac{1}{mn}\|S-S_0\|_F^2, \tag{3-1}$$

which is the sum of all squared pixel value difference [34]. Based on MSE, which calculates the power of noises that affect the recovery fidelity, PSNR calculates the ratio between maximum possible power of signal and the power of noise,

$$\text{PSNR } = 10\log_{10}(\frac{\text{MAX}^2}{\text{MSE}}), \tag{3-2}$$

where MAX is the maximum pixel value of a 8-bit image which is 255.

3-2-2 F-measure

As suggested and adopted in [29], [35] and [36], F-measure is used to evaluate the foreground extraction result of algorithms such as TVRPCA, DECOLOR and GRASTA in these papers. F-measure is a combination of two evaluation methods of test accuracy, precision p and recall r [37]. The result of assigned classification is known as true and false. True means correct classification while false means incorrect. In the experiment, the decomposition result is first transformed into a matrix with 255 and 0 only. All the positive-value pixels are turned into 255 and the transformed matrix is then compared with ground truth data which only contains 255 and 0 as well. If the pixels at the same position in these two matrices have the same value, the classification result is true. Positive or negative means assignment to positive or negative category [38]. In the ground truth data matrix, pixel with value of 255 is positive while pixel with value of 0 is negative. Therefore there are in total four states of the classification result, True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Only TP and TF are correct classification result. FP incorrectly identifies negative result and FN incorrectly rejects positive result. Precision p is defined as $\frac{TP}{TP+FP}$ which is the ratio of correct positive results to all positive results no matter true or false. Recall r is defined as $\frac{TP}{TP+FN}$ which is the correct positive results over all the results that should have been identified as positive. Based on these two evaluation methods, F-measure then combines them as:

$$\text{F-measure} = 2\frac{\text{p}\cdot\text{r}}{\text{p}+\text{r}}. \tag{3-3}$$

It is actually harmonic mean between precision and recall. It averages both of them to evaluate the classification result. Additional weight can also be put on p and r to emphasize the importance of either p or r.

3-2-3 ROC curve

Receiver Operating Characteristic (ROC) curve depicts the relation between True Positive Rate (TPR) and False Positive Rate (FPR) at different thresholds. True Positive Rate is

$\frac{TP}{TP+FN}$ which is recall defined in F-measure and False Positive Rate is $\frac{FP}{FP+TN}$. In the experiment, all the classification results are rescaled into the range of 0 to 1. Each time it chose one result as threshold. The result above this threshold is positive which in the experiment is classified as foreground pixel. It calculates TPR and FPR for every threshold and every result will be a dot on the curve. At last all the dots are connected to obtain ROC curve. The point (0,1) on the top left corner represents error free case where it gives perfect classification while the diagonal line from bottom left to top right represents random guess case. Area Under Curve (AUC) is the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample [38].

3-3 Parameter selection

In algorithm 1, D is the data matrix which is given and reshaped. The results of L and S from IALM algorithm can be chosen as the initial values L_0 and S_0 for algorithm 1. The choice of λ and μ first follows the recommendation in paper [9], [26] and [27]. But when implementing the algorithm, the result of S is not as expected where it is not fully nonnegative or extremely sparse that does not show the foreground object. The reason why S is not fully nonnegative lies in the choice of initial parameters especially λ and number of iterations. Since in the update rules of algorithm 1, λ acts as a threshold $\frac{\lambda}{2\mu}$ to only keep the values above this threshold in S. If the threshold is too small it cannot fully eliminate negative elements in S when reaching the maximum number of iterations that has been set beforehand. More iterations and time will be needed to remove those negative elements. And if λ is chosen too large then every element in S cannot exceed the threshold and become zero. That is the reason why S is extremely sparse. Thus appropriate λ should provide the balance between achieving nonnegativity within given iteration and suitable sparsity to display the foreground object.

Then a test is made in order to choose λ for proposed algorithm 1 that gives both nonnegative and sparse S within maximum number of iterations. Considering the iteration time, the test used 10 frames of bottle video sequence which is in the dimension of $R^{68096\times10}$. The test chose λ in the range of λ_0 to $4\lambda_0$ where λ_0 is the initial valued defined as $\frac{1}{\sqrt{max(m,n)}}$. And the maximum iteration is 10000 after several trials. The sparsity and number of negative elements of matrix S are shown in Figure 3-1 and Figure 3-2.

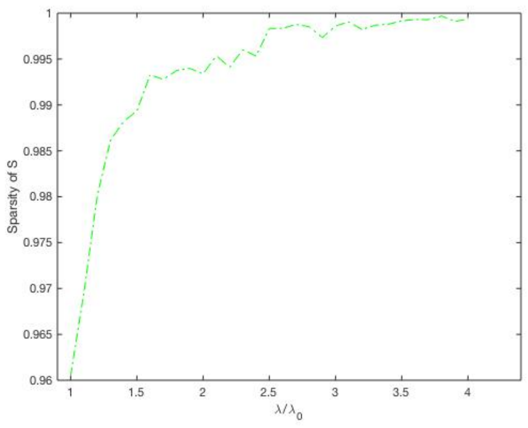

Figure 3-1: Sparsity of S as λ changes

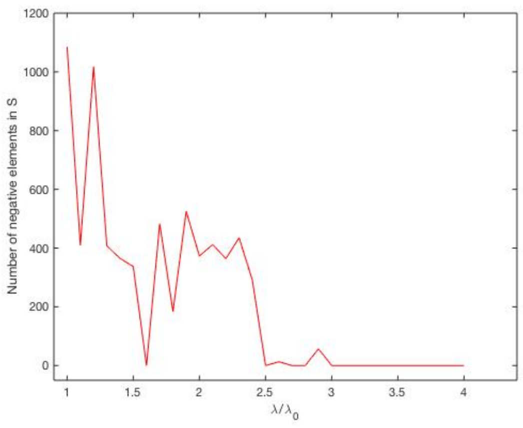

Figure 3-2: Number of negative elements in S as λ changes

Sparsity is calculated as one minus the ratio between number of nonnegative elements and number of all the elements in S by using **nnz** command, which calculates the number of nonzero elements in a matrix. As λ increases sparsity of S increases as well and it becomes almost completely zero matrix if λ is chosen too large as explained before. And for the number of negative elements in S, if λ is chosen too small such as smaller than $2\lambda_0$, negative elements cannot be fully eliminated within 10000 iterations. But if λ is for example $3\lambda_0$, those negative elements can be fully eliminated with around 6000 iterations which is much faster.
Figure 3-3 shows the number of negative elements in S with the change of λ in a more compact way by changing y axis to logarithmic scale. The break between two consecutive points is when the number of negative elements is zero since logarithmic scale only shows positive numbers.

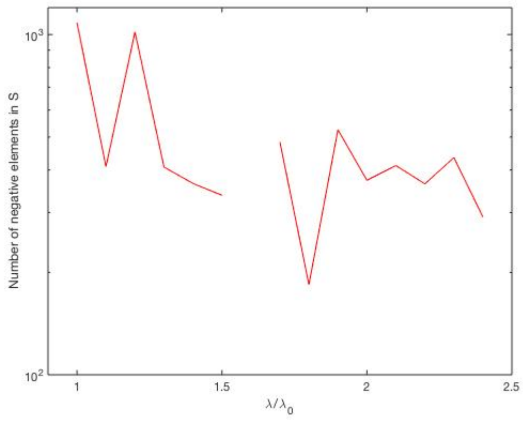

Figure 3-3: Number of negative elements in S as λ changes in log scale

In the test above the initial values L_0 and S_0 does not have much change because λ remains the same for IALM algorithm. Then the same change for λ is also applied to IALM to obtain different initial values L_0 and S_0. The result shows that the negative elements are fully eliminated in less iterations as shown in Figure 3-4.

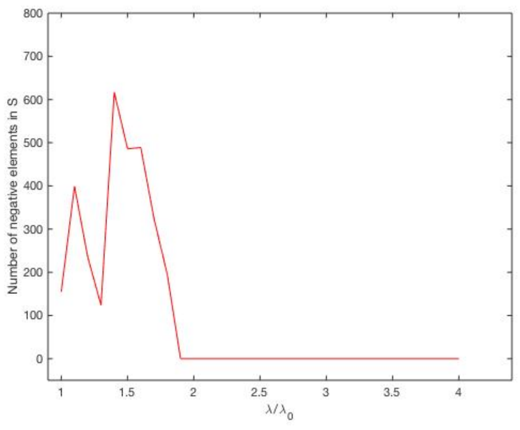

Figure 3-4: Number of negative elements in S as λ changes test 2

The negative elements in S are fully removed with around 5000 iterations when λ is $2\lambda_0$. Based on the test to choose the most proper λ considering the number of iterations to clear the negative elements and the sparsity of in S, the range between $2\lambda_0$ to $3\lambda_0$ provides the expected result. Thus in the experiment λ is firstly chosen to be $2\lambda_0$ and it also uses $2\lambda_0$ to obtain the initial values L_0 and S_0 from IALM algorithm. And λ can be adjusted around this range to get more satisfying result. The same test is then also made to find the most suitable λ range for proposed algorithm 2. It finds out that the most suitable λ lies in the range between λ_0 and $2\lambda_0$. In the experiment, there is no need to keep the same λ for every algorithm since the most suitable λ range is different for every algorithm.

3-4 Algorithm result comparison

The two newly proposed algorithms in chapter 2 will be compared together with RPCA algorithm in the experiment. Three video sequences bottle, surfers and boats were selected from the dataset. They are all photos taken on the sea surface where the moving foreground object is bottle, surfer and boat respectively. The size of bottle video sequence is $224\times304\times31$. The size of surfers video sequence is $224 \times 344 \times 41$ and the size of boats video sequence is $224 \times 344 \times 31$. From each video sequence one frame is taken out and shown in Figure 3-5. The number of frames used in the experiment were chosen after using different number of frames and comparing the total iteration time. The time for running 10 frames was around 12 minutes. The result for a single frame does not change much except the total running time increases to a large extent with the increase of frame number. Considering the total iteration time, only the first 10 frames from each dataset were used in the experiment.

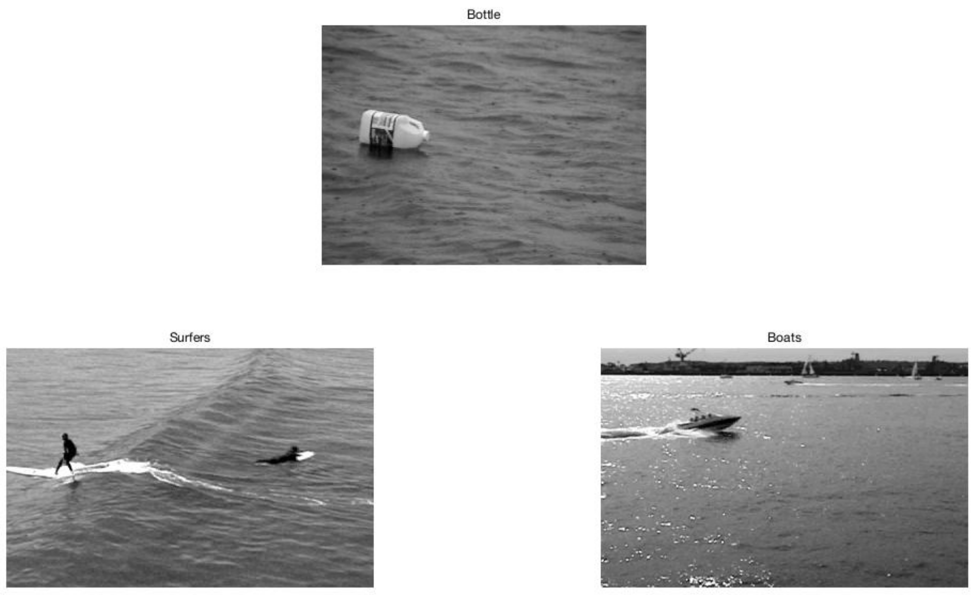

Figure 3-5: Video frames from three videos sequences

Firstly, the recovered S of these three video sequences will be shown directly to compare their algorithm performance in a qualitative way. Then MSE, PSNR, F-measure and ROC of proposed algorithms together with RPCA will give a quantitative analysis of the result.

3-4-1 Qualitative analysis

- **Bottle sequence:** Figure 3-6 shows the recovered moving bottle with ground truth data. Compared to the other two video sequence, bottle video sequence is comparatively easier to recover since the moving object is obvious in the frame and it is the only object that needs to be extracted in the frame. One difficulty in this scene is that the background is moving water surface which can cause false detection. The false detection of moving background is more obvious in Figure 3-7 if the positive pixels are all set to 255 even though some pixel values are extremely small values. Red area is False Positive. Green area is True Positive while blue is False Negative. Proposed algorithm 1 and especially proposed algorithm 2 not only detects the moving object but treat some background pixels as foreground as well. Some small positive values are kept instead of being treated as zero. It will be further explained in quantitative analysis.

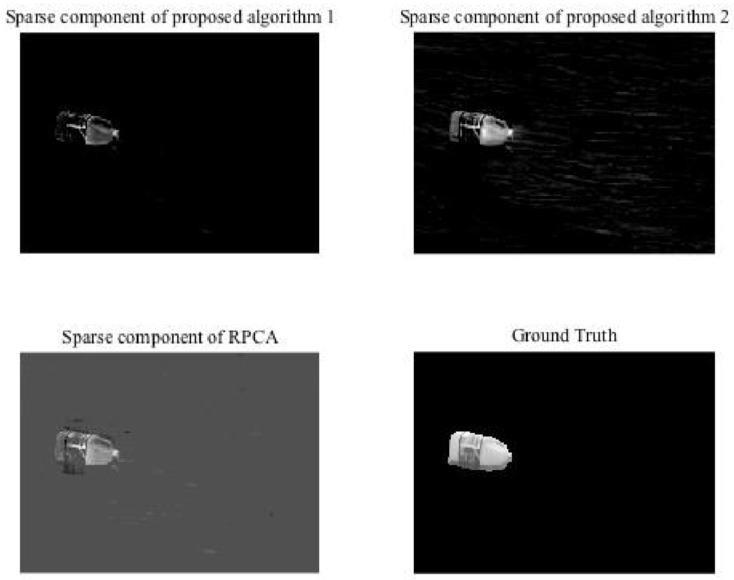
Sparse component of proposed algorithm 1
Sparse component of proposed algorithm 2
Sparse component of RPCA
Ground Truth

Figure 3-6: Sparse component of proposed algorithm 1 (upper left), proposed algorithm 2 (upper right), RPCA (bottom left) and ground truth data (bottom right)

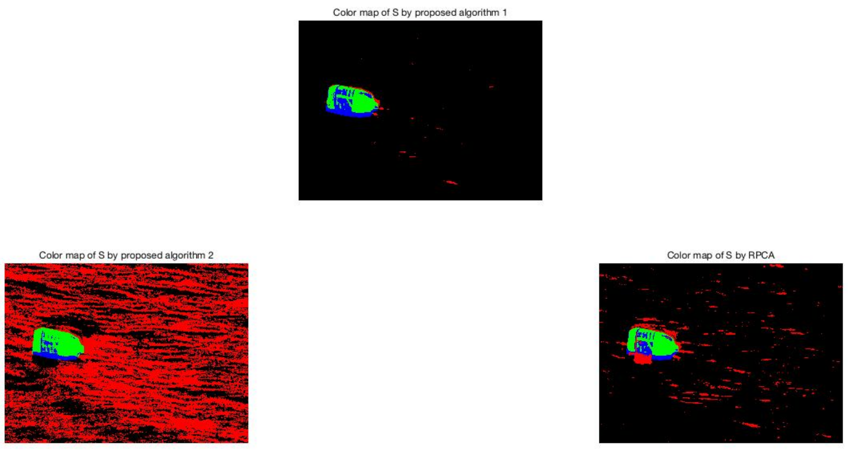
Color map of S by proposed algorithm 1
Color map of S by proposed algorithm 2
Color map of S by RPCA

Figure 3-7: Color maps of proposed algorithm 1 (middle), proposed algorithm 2 (bottom left) and RPCA (bottom right)

- **Surfers sequence:** As shown in Figure 3-8, surfers video sequence is more difficult to recover since the water surface keeps waving with the moving object. All the algorithms have false detection that treat some of background scene as foreground object. The false detection is more obvious in Figure 3-9. Especially for algorithm 2, even though the green area is comparatively large but the color map also has many false positive pixels.

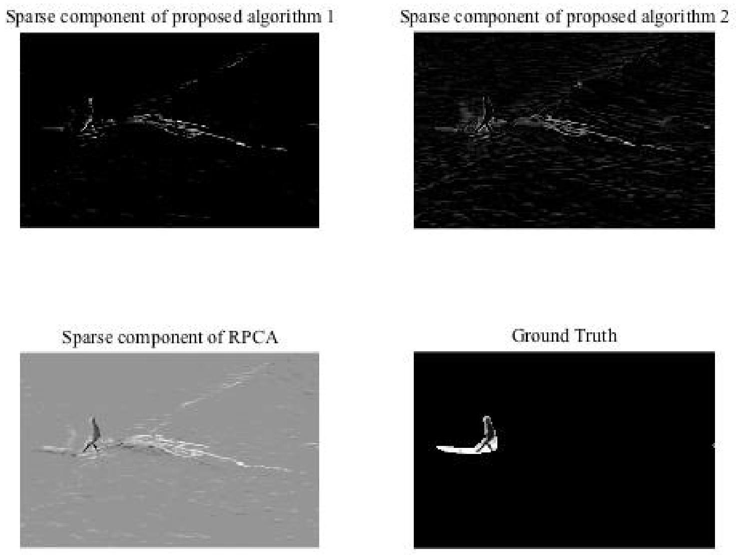
Sparse component of proposed algorithm 1
Sparse component of proposed algorithm 2
Sparse component of RPCA
Ground Truth

Figure 3-8: Sparse component of proposed algorithm 1 (upper left), proposed algorithm 2 (upper right), RPCA (bottom left) and ground truth data (bottom right)

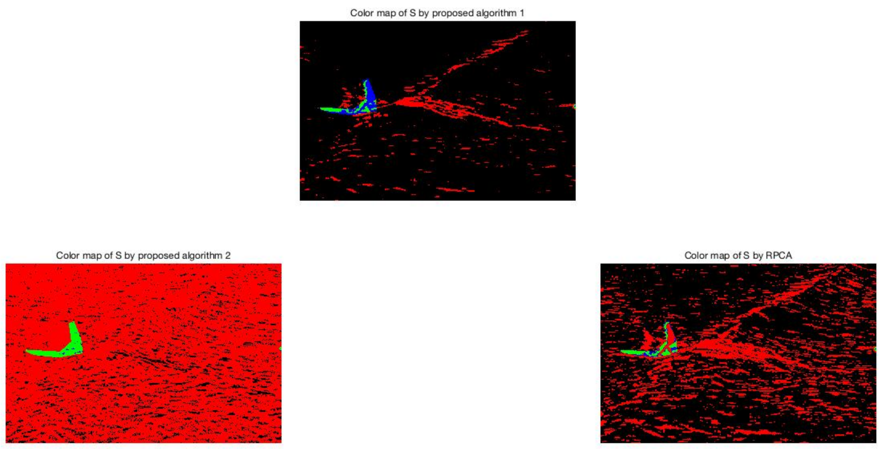
Color map of S by proposed algorithm 1
Color map of S by proposed algorithm 2
Color map of S by RPCA

Figure 3-9: Color maps of proposed algorithm 1 (middle), proposed algorithm 2 (bottom left) and RPCA (bottom right)

- **Boats sequence:** Figure 3-10 shows the same difficulty as surfers sequence. It also treats some of the background scene as foreground.

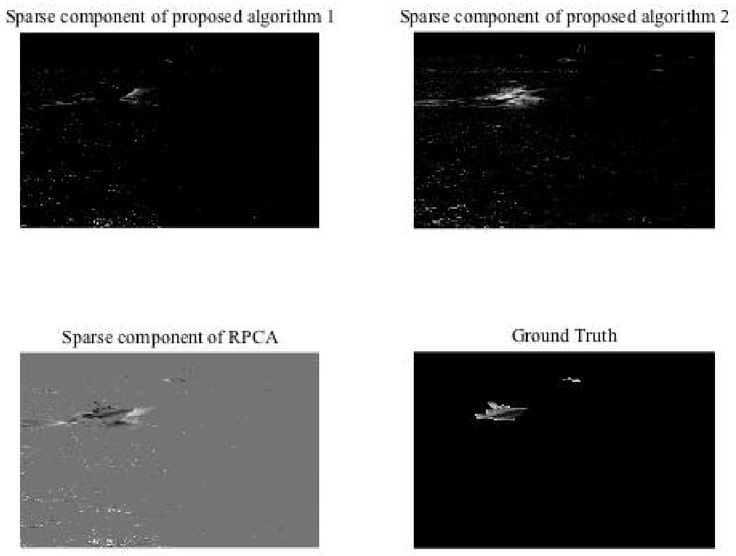
Sparse component of proposed algorithm 1
Sparse component of proposed algorithm 2
Sparse component of RPCA
Ground Truth

Figure 3-10: Sparse component of proposed algorithm 1 (upper left), proposed algorithm 2 (upper right), RPCA (bottom left) and ground truth data (bottom right)

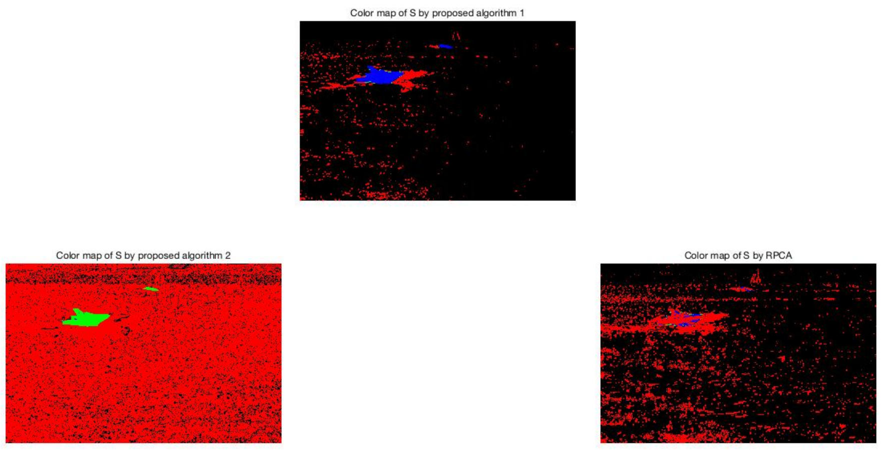
Color map of S by proposed algorithm 1
Color map of S by proposed algorithm 2
Color map of S by RPCA

Figure 3-11: Color maps of proposed algorithm 1 (middle), proposed algorithm 2 (bottom left) and RPCA (bottom right)

3-4-2 Quantitative analysis

Besides the qualitative analysis of whether the result shows the foreground object clearly or not, performance measures give a comprehensive evaluation of the results as well. Figure 3-12, 3-13 and 3-14 show the ROC curve for the three video sequences. Green curve represents RPCA. Blue curve is proposed algorithm 1 and red curve is Proposed algorithm 2. As introduced before, horizontal axis x of ROC is False Positive Rate while vertical axis y is True Positive Rate. Higher TPR will move the curve up along y axis and thus give larger AUC.

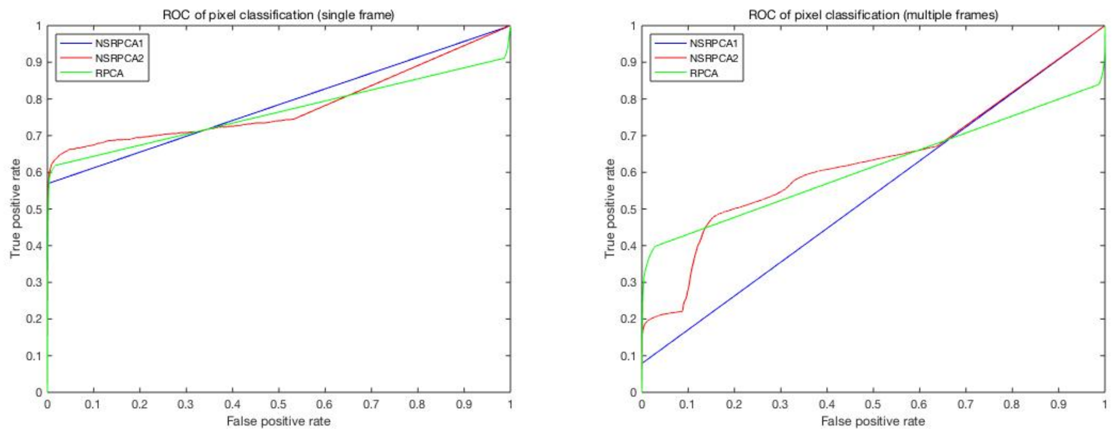

Figure 3-12: Single frame (left) and multiple frames (right) ROC curve of bottle sequence

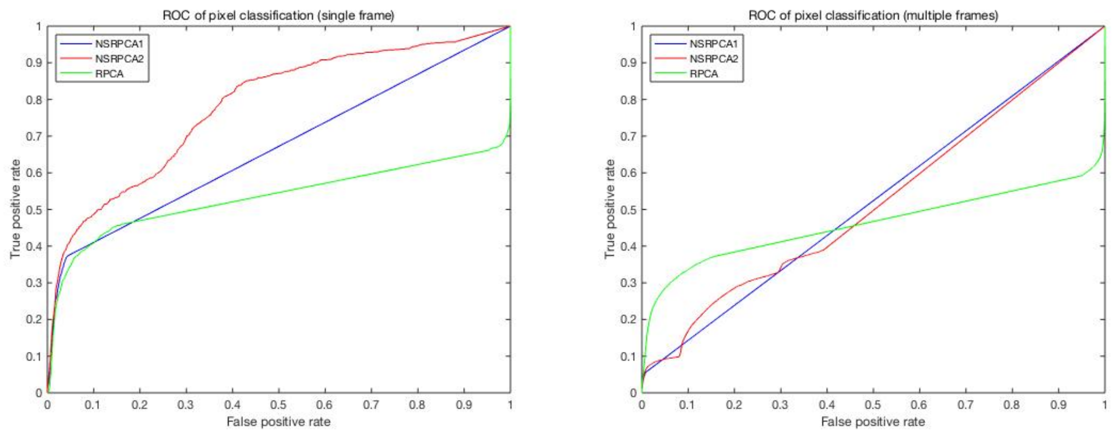

Figure 3-13: Single frame (left) and multiple frames (right) ROC curve of surfers sequence

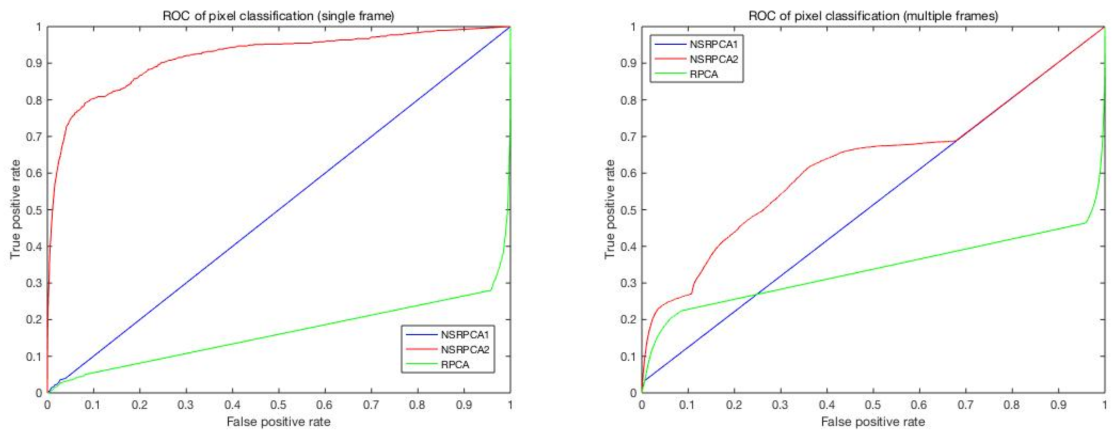

Figure 3-14: Single frame (left) and multiple frames (right) ROC curve of boats sequence

AUC	Algorithm 1		Algorithm 2		RPCA	
	Single frame	Multiple frames	Single frame	Multiple frames	Single frame	Multiple frames
Bottle	**0.7893**	0.5393	0.7809	**0.6372**	0.7640	0.6143
Surfers	0.6673	**0.5240**	**0.7821**	0.5205	0.5335	0.4600
Boats	0.5004	0.5137	**0.9174**	**0.6264**	0.1638	0.3367

Table 3-1: AUC of single frame and multiple frames

Table 3-1 shows AUC for the three videos sequences. In general algorithm 1 and 2 work better than RPCA both for single frame and multiple frames. Algorithm 2 works significantly well for boats sequence with 0.9174 true positive rate for single frame while the other two algorithms have a relatively low true positive rate. Algorithm 2 distinguishes the foreground pixel correctly even when background has constant changes. When it comes to the performance for single and multiple frames, the three algorithms work better for single frame because for multiple frames both the foreground and background scene move which makes it difficult to sort out the moving object. Thus it decreases the true positive rate.
But when comparing the classification precision and F-measure score in Table 3-2 and Table 3-3, algorithm 2 does not show any superiority than the other two algorithms. Instead algorithm 1 and RPCA has higher precision and F-measure score. The reason lies in the calculation difference between ROC and precision. For ROC, it uses every pixel value as the classification threshold. Some pixel values are positive but are not classified as foreground because it is below the current classification threshold. But for precision, all the positive values are set to 255 even when it is extremely small. Under this condition, the small pixel values are all classified as foreground pixels which decrease the precision to a large extent. It does not mean algorithm 2 performs even worse than the other two algorithms since it has higher recall than the other two algorithms. Higher recall means it has less false negative elements. Algorithm 2 classifies the pixels that should have been classified as positive more accurately than the other two algorithms. Besides, in Table 3-4 and Table 3-5, algorithm 2 has lower MSE than the other two algorithms and its PSNR is higher than the other two. It only shows that algorithm 2 does not abandon some background pixels even when it is small pixel value. This is due to the fact that algorithm 2 does not approximate foreground pixel like algorithm 1 does. Algorithm 1 has higher precision and F-measure score than algorithm 2 because it approximates those small pixel values as zero while in algorithm 2 those small pixel values are retained.

Single frame	Algorithm 1			Algorithm 2			RPCA		
	p	r	F-measure	p	r	F-measure	p	r	F-measure
Bottle	**0.8444**	0.5703	**0.6808**	0.0410	**0.7474**	0.0778	0.5438	0.7079	0.6151
Surfers	**0.0951**	0.3730	**0.1515**	0.0130	**0.9589**	0.0256	0.0573	0.7978	0.1070
Boats	0.0105	0.0389	0.0165	0.0118	**0.9899**	0.0234	**0.0854**	0.7714	**0.1537**
Average	**0.3167**	0.3274	0.2829	0.0219	**0.8987**	0.0423	0.2288	0.7590	**0.2919**

Table 3-2: Precision, Recall and F-measure of single frame performance

Multiple frames	Algorithm 1			Algorithm 2			RPCA		
	p	r	F-measure	p	r	F-measure	p	r	F-measure
Bottle	**0.7782**	0.0793	0.1439	0.0300	**0.6710**	0.0574	0.3696	0.5586	**0.4448**
Surfers	**0.1164**	0.0556	0.0753	0.0182	0.3887	0.0347	0.0841	**0.7801**	**0.1518**
Boats	0.0429	0.0359	0.0391	0.0108	0.6876	0.0212	**0.0864**	**0.7606**	**0.1552**
Average	**0.3152**	0.0569	0.0861	0.0196	0.5824	0.0378	0.1800	**0.6998**	**0.2506**

Table 3-3: Precision, Recall and F-measure of multiple frames performance

Single frame	Algorithm 1		Algorithm 2		RPCA	
	MSE	PSNR	MSE	PSNR	MSE	PSNR
Bottle	1.7879×10^3	15.6073	$\mathbf{1.5486 \times 10^3}$	16.2315	1.7572×10^3	15.6826
Surfers	**756.6485**	19.3419	779.9296	19.2102	910.0455	18.5402
Boats	697.0696	19.6980	**671.4456**	19.8607	887.1388	18.6509
Average	1080.5393	18.2157	**999.9917**	18.4341	1184.7948	17.6246

Table 3-4: MSE and PSNR of single frame performance

Multiple frames	Algorithm 1		Algorithm 2		RPCA	
	MSE	PSNR	MSE	PSNR	MSE	PSNR
Bottle	1.8371×10^3	15.4895	$\mathbf{1.7684 \times 10^3}$	15.6551	1.8098×10^3	15.5545
Surfers	1.1802×10^3	17.4112	$\mathbf{1.1768 \times 10^3}$	17.4240	1.3290×10^3	16.8956
Boats	692.4874	19.7267	**685.1051**	19.7732	812.9658	19.0301
Average	1236.5958	17.5425	**1210.1017**	17.6174	1317.2558	17.1601

Table 3-5: MSE and PSNR of multiple frames performance

Chapter 4

Conclusion

In this chapter the thesis work will be first summarized. The advantages and disadvantages of the proposed algorithms will be discussed. Based on the analysis, possible improvements and expectations of the algorithms will be mentioned.

- **Conclusion:** The main concern of this thesis is to deal with the negative pixels in the result of background and foreground decomposition since negative pixels are not visible in images and the foreground object that is composed of negative pixels can thus be not visible in images. Then improvement is made on the basis of Robust Principal Component Analysis (RPCA), which is the main algorithm in the application of foreground extraction. With the help of another nonnegative decomposition algorithm Nonnegative Matrix Factorization (NMF), this thesis proposes two algorithms that decompose the sparse component with nonnegative constraint. Test shows that given enough iterations, proposed algorithms can fully eliminate the negative elements in sparse component. These two algorithms are further tested to extract the foreground object from chosen dataset. Both qualitative and quantitative comparisons like precision, F-measure and ROC are made between these two algorithms and RPCA algorithm. The comparison shows the proposed algorithms have higher true positive rate in terms of recovering the video frames in the dataset.
- **Comparison:** Clearly the advantanges of RPCA lies in its fast iteration compared to the other two algorithms. RPCA takes 104, 236 and 256 iterations to run the three video sequences respectively. On the basis of RPCA algorithm, proposed algorithm 1 and 2 add additional nonnegative constraint to the optimization problem. Because of this constraint, it further complicates the update rules where there are gradients U_S and V_S that need to be computed in every iteration. Even when the change of U_S and V_S are relatively small between two successive iterations, the whole iteration will keep going once the nonnegative constraint is not fulfilled. Thus it leads to iterations that reach to 5000 for proposed algorithm 1 and 2.
  But the proposed algorithms improve the quality of extracted foreground object at the

cost of more iterations. Proposed algorithms first fulfills the nonnegative constraint that RPCA cannot achieve. Foreground pixels are all nonnegative as verified in the experiment of parameter selection with the help of nonnegative constraint that NMF provides. Second, according to the analysis the true positive rate of proposed algorithm 1 and 2 is higher than that of PRCA. Algorithm 1 and RPCA perform better than algorithm 2 in terms of precision and F-measure score especially for multiple frames. The recall of algorithm 2 outperforms the other two algorithms which means it has less false negative pixels. And the relative error with ground truth data of proposed algorithm 1 and 2 is smaller than that of RPCA. The result shows the proposed algorithms recover the data with smaller error and algorithm 2 does not lose foreground information as much as RPCA does.

- **Possible improvements:** In terms of the algorithm itself, the update rules have some gradient-related updates. The gradient update of matrices can complicate the whole update rules. Except for the update of L, the algorithm uses gradient of the corresponding cost function to update S, U_S and V_S. There can be better ways to solve the optimization problem than taking the derivatives of these gradients. Besides, the whole iteration keeps running even when the relative error between the recovered matrices and original data is relatively small because nonnegativity constraint is not fulfilled. Improvement can also be made to incorporate the nonnegative constraint in the cost function in a better way.
  And because of the nonnegative constraint is not fulfilled, the running time of the proposed algorithm for 10 frames in the dataset is approximately 12 minutes. The computation of gradient and the fulfillment of nonnegative constraint consume much time in the whole algorithm. To figure out how to reduce the total time consumption is also another improvement of the proposed algorithm.
  As analysed in the experiment, the precision of proposed algorithm 2 needs to be further improved. The foreground object is extracted with high true positive rate but it also involves small pixel values. There can be a better way to deal with those small background pixel values such as turn them all into zero and thus increase the precision. In the thesis only the extracted sparse component is compared and the quality of it is analyzed. The next possible step can focus on low-rank component since it is sometimes dynamic and difficult to extract. As nonnegative low-rank component RPCA suggests, experiments can be further made in background extraction.

- **Future work:** This thesis combines the basic models of RPCA and NMF. The proposed algorithm uses the model of RPCA and nonnegative features of NMF, which shows the possibility of combing these two basic algorithms. In addition to the proposed algorithm, there can be more possibilities using other existing models of RPCA and NMF such as the models used in the applications in chapter 1 and also examine their feasibility in terms of giving nonnegative pixels. There also exist other algorithms that guarantee the nonnegativity of the decomposed products beside NMF, which can be seen as a substitute of NMF algorithm.
  Beside discovering the feasibility of other algorithms, the proposed algorithm can also be used in other fields. For example, the sparse component sometimes needs to be removed from original data. The proposed algorithm can not only extract but discard the sparse information as well. And the result of the proposed algorithm, which is

the foreground object, can be collected for further use such as object identification and classification. Object identification can then be used in various environment like road, water surface and sky. Applications such as automated driving system can use the result of the algorithm to avoid collision with other moving objects. Other possible post-processing can also be made on the result of the proposed algorithm.

Appendix A

Matlab codes of proposed algorithms

A-1 Main files

A-1-1 Demo

```
clc
clear

% load image
myFolder = '/Users/lingchenyang/Documents/MATLAB/Thesis/NRPCA/bottle';
myFile = fullfile(myFolder, '*.jpg');
myImage = dir(myFile);
files = {myImage.name};
files = natsortfiles(files);
for k = 1:numel(files)
  Im{k} = imread(fullfile(myFolder,files{k}));
  data(:,k) = double(reshape(Im{k},[],1));
end

% load ground truth data
load bottle_GT.mat %or boats_GT, surfers_GT

%% Proposed algorithm 1 & 2
D = data(:,1:10);
%D = reshape(D,[32 32 3]);
%D = rgb2gray(D);
%D = double(D);

% size of the image
n_1 = size(D,1);
n_2 = size(D,2);

% parameter definition
```

```
%i = 1;
%for coefficient_lambda = 1:0.1:4
coefficient_lambda = 2;
opts.lambda = 1/sqrt(max(n_1,n_2))*coefficient_lambda;
opts.mu = n_1*n_2/(4*norm(D,1));
opts.rho_D = 1e-6;
opts.rho_S = 1e-6;
opts.rho = 1e-6;
opts.max_iter = 10000;
[L0,S0,sparsity_S0,iter0] = basic_RPCA(D,opts);
opts.L0 = L0;
opts.S0 = S0;
coefficient_lambda = 2.5;
opts.lambda = 1/sqrt(max(n_1,n_2))*coefficient_lambda;
tic
[L,S,U_S,V_S,count_L,count_S,rank_L,sparsity_S,error_D,error_S,size_k,
    iter] = NSRPCA(D,opts);
coefficient_lambda = 1;
opts.lambda = 1/sqrt(max(n_1,n_2))*coefficient_lambda;
[L1,S1,count_L1,count_S1,count_U_S1,count_V_S1,rank_L1,sparsity_S1,
    error_D1,size_k1,iter1] = NSRPCA_no_S(D,opts);
toc

%negative_num_S(i) = numel(find(S<0));
%sparsity_S(i) = sparsity_S(size_k);
%iter_S(i) = iter;
%i = i+1;
%end

%%
% Display of L, S, D and GT data
k = 1;
figure(1)
imshow(reshape(L(:,k),size(GT(:,:,k))),[])
title('Low-rank component','Color','k','FontName','Times New Roman','
    FontSize',12)

%%
figure(2)
imshow(reshape(S(:,k),size(GT(:,:,k))),[])
title('Sparse component','Color','k','FontName','Times New Roman','
    FontSize',12)

%%
figure(3)
imshow(reshape(D(:,k),size(GT(:,:,k))),[])
title('Original image','Color','k','FontName','Times New Roman','FontSize
    ',12)

%%
figure(4)
%imshow(GT(:,:,k))
imshow(GT(:,:,k).*double(Im{k}),[])
```

```
title('Ground truth','Color','k','FontName','Times New Roman','FontSize'
    ,12)

%%
figure(5)
imshow(reshape(S0(:,k),size(GT(:,:,k))),[])
title('S0','Color','k','FontName','Times New Roman','FontSize',12)

%%
% Display L, S, D and GT data
k = 1;
figure(6)
subplot(2,2,1);
imshow(reshape(L(:,k),size(GT(:,:,k))),[])
title('Low-rank component of proposed algorithm 1','FontName','Times New
    Roman','Color','k','FontSize',12);

subplot(2,2,2);
imshow(reshape(S(:,k),size(GT(:,:,k))),[])
title('Sparse component of proposed algorithm 1','FontName','Times New
    Roman','Color','k','FontSize',12);

subplot(2,2,3);
imshow(reshape(D(:,k),size(GT(:,:,k))),[])
title('Original image','FontName','Times New Roman','Color','k','FontSize
    ',12);

subplot(2,2,4);
%imshow(GT(:,:,k),[]);
imshow(GT(:,:,k).*double(Im{k}),[]);
title('Ground Truth','FontName','Times New Roman','Color','k','FontSize'
    ,12);

%%
% Display L1, S1, D and GT data
k = 1;
figure(7)
subplot(2,2,1);
imshow(reshape(L1(:,k),size(GT(:,:,k))),[])
title('Low-rank component of proposed algorithm 2','FontName','Times New
    Roman','Color','k','FontSize',12);

subplot(2,2,2);
imshow(reshape(S1(:,k),size(GT(:,:,k))),[])
title('Sparse component of proposed algorithm 2','FontName','Times New
    Roman','Color','k','FontSize',12);

subplot(2,2,3);
imshow(reshape(D(:,k),size(GT(:,:,k))),[])
title('Original image','FontName','Times New Roman','Color','k','FontSize
    ',12);

subplot(2,2,4);
```

```
%imshow(GT(:,:,k))
imshow(GT(:,:,k).*double(Im{k}),[]);
title('Ground Truth','FontName','Times New Roman','Color','k','FontSize'
    ,12);

%%
% S component of proposed algorithm 1, algorithm 2, RPCA and GT data
k = 1;
figure(8)
subplot(2,2,1);
imshow(reshape(S(:,k),size(GT(:,:,k))),[])
title('Sparse component of proposed algorithm 1','FontName','Times New
    Roman','Color','k','FontSize',12);

subplot(2,2,2)
imshow(reshape(S1(:,k),size(GT(:,:,k))),[])
title('Sparse component of proposed algorithm 2','Color','k','FontName','
    Times New Roman','FontSize',12)

subplot(2,2,3);
imshow(reshape(S0(:,k),size(GT(:,:,k))),[])
title('Sparse component of RPCA','Color','k','FontName','Times New Roman'
    ,'FontSize',12)

subplot(2,2,4)
%imshow(GT(:,:,k),[]);
imshow(GT(:,:,k).*double(Im{k}),[]);
title('Ground Truth','FontName','Times New Roman','Color','k','FontSize'
    ,12);

%% ROC curve of single and multiple frames of proposed algorithm 1 and 2

[X_multi,Y_multi,AUC_multi] = ROC_curve_multi(S,GT);
[X1_multi,Y1_multi,AUC1_multi] = ROC_curve_multi(S1,GT);
[X0_multi,Y0_multi,AUC0_multi] = ROC_curve_multi(S0,GT);

[X_single,Y_single,AUC_single] = ROC_curve_single(S,GT,k);
[X1_single,Y1_single,AUC1_single] = ROC_curve_single(S1,GT,k);
[X0_single,Y0_single,AUC0_single] = ROC_curve_single(S0,GT,k);

figure(9)
plot(X_multi,Y_multi,'b')
hold on
plot(X1_multi,Y1_multi,'r')
plot(X0_multi,Y0_multi,'g')
hold off
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC of pixel classification (multiple frames)')
legend('NSRPCA1','NSRPCA2','RPCA','Location','northwest')

figure(10)
plot(X_single,Y_single,'b')
```

```
hold on
plot(X1_single,Y1_single,'r')
plot(X0_single,Y0_single,'g')
hold off
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC of pixel classification (single frame)')
legend('NSRPCA1','NSRPCA2','RPCA','Location','northwest')

%% Direct colormap of True Positive, False Positive, False Negative
    pixels
[TP_S,FP_S,TN_S,FN_S,S_color] = colormap(S,GT,k);figure(11);imshow(
    S_color);title('Color map of S by proposed algorithm 1');
%%
[TP_S1,FP_S1,TN_S1,FN_S1,S1_color] = colormap(S1,GT,k);figure(12);imshow(
    S1_color);title('Color map of S by proposed algorithm 2');
%%
[TP_S0,FP_S0,TN_S0,FN_S0,S0_color] = colormap(S0,GT,k);figure(13);imshow(
    S0_color);title('Color map of S by RPCA');

%% Peak Signal-to-noise ratio of single and multiple frames
% GT_re = reshape(GT,[],size(GT,3));
% GT_re = D.*GT_re(:,1:10);
% GT_re = reshape(GT_re,size(GT,1),size(GT,2),[]);

[MSE_multi, ratio_multi] = PSNR_multi(S,GT);
[MSE1_multi, ratio1_multi] = PSNR_multi(S1,GT);
[MSE0_multi, ratio0_multi] = PSNR_multi(S0,GT);

[MSE_single, ratio_single] = PSNR_single(S,GT,k);
[MSE1_single, ratio1_single] = PSNR_single(S1,GT,k);
[MSE0_single, ratio0_single] = PSNR_single(S0,GT,k);

%% F-measure of single and multiple frames
[p_S_multi, r_S_multi, score_S_multi] = F_measure_multi(S,GT);
[p_S_single, r_S_single, score_S_single] = F_measure_single(S,GT,k);

[p_S1_multi, r_S1_multi, score_S1_multi] = F_measure_multi(S1,GT);
[p_S1_single, r_S1_single, score_S1_single] = F_measure_single(S1,GT,k);

[p_S0_multi, r_S0_multi, score_S0_multi] = F_measure_multi(S0,GT);
[p_S0_single, r_S0_single, score_S0_single] = F_measure_single(S0,GT,k);

%% negatie number, sparsity and iteration number plot
% figure(14)
% plot(negative_num_S,'-r');
% %set(gca, 'YScale', 'log');
% set(gca, 'YLim', [-50 800]);
% xticks(1:5:31)
% xticklabels(1:0.5:4)
% xlabel('\lambda/\lambda_0');
% ylabel('Number of negative elements in S1');
%
```

```
% figure(15)
% plot(1-sparsity_S,'-.g');
% xticks(1:5:31)
% xticklabels(1:0.5:4)
% xlabel('\lambda/\lambda_0');
% ylabel('Sparsity of S1');
%
% figure(16)
% plot(iter_S,'-*b');
% set(gca, 'YLim', [-100 11000]);
% xticks(1:5:31)
% xticklabels(1:0.5:4)
% xlabel('\lambda/\lambda_0');
% ylabel('Iteration number');

%% resize axis
% fig = openfig('negative_num_lambda.fig');
% set(gca, 'YLim', [-50 1200]);

%% plot error_D ,error_S and rank_L with respect to rank change
% figure()
% plot(error_D,'-r');
% %axis([0 3000 0 0.001])
% axis([1 224 0 0.0001]);
% hold on
% plot(error_S,'.b');
% hold off
% legend('error_D','error_S');
%
% figure()
% plot(rank_L,'-.g');
% xlim([1 224]);
% xlabel('Size of U_S and V_S');
% ylabel('Rank of L');
```

A-1-2 Proposed algorithm 1

```
function [L,S,U_S,V_S,count_L_final,count_S_final,rank_L,sparsity_S,
    error_D_final,error_S_final,k,iter] = NSRPCA(D,opts)
%  Proposed algorithm 1: Nonnegative Sparse Component RPCA 1
%  cost function L(L,S,U_S,V_S) =
%  ||L||_*+\lambda||S||_1+<Y_D,D-L-S>+\mu/2||D-L-S||_F^2+...
%  <Y_S,S-U_S*V_S>+\mu/2||S-U_S*V_S||_F^2+...
%  \mu/2||U_S-C_1||_F^2+\mu/2||V_S-C_2||_F^2
%  constraints: D = L+S
%               S = U_S*V_S
%               U_S>=0, V_S>=0

%   input: D is the image matrix
%          opts:
%          opts.lambda is the coefficient of sparse component
%          opts.mu is the coefficient of error term in the cost function
%          opts.max_iter is the maximum iteration number
%           opts.rho_D is one of the thresholds in the stopping criterion
%           opts.rho_S is one of the thresholds in the stopping criterion
%           opts.L0 is the initial low-rank component
%           opts.S0 is the initial sparse component
%           opts.Y_D is the initial Lagrange multiplier


[lambda,mu,max_iter,rho_D,rho_S,L0,S0,Y_D] = parseInputs(D,opts);
for k = 2
    L = L0;
    S = S0;
    Y_S = rand(size(S));
    n_1 = size(D,1);
    n_2 = size(D,2);

    U_S = rand(size(S,1),k);
    V_S = rand(k,size(S,2));
    C_1 = 0;
    C_2 = 0;

    %rank_L = zeros(max_iter,1);
    %error_D = zeros(max_iter,1);
    %error_S = zeros(max_iter,1);

    for iter = 1:max_iter
        % update L
        [U,Sigma,V] = svd(D+Y_D/mu-S,'econ');
        Sigma1 = sign(Sigma) .* max(abs(Sigma) - 1/mu, 0);
        L = U*Sigma1*V';
        %rank_L(iter) = rank(L);

        % update S
        S = sign(-(L-D+Y_S/mu-Y_D/mu-U_S*V_S)/2) .* max(abs(-(L-D+Y_S/mu-
            Y_D/mu-U_S*V_S)/2) - lambda/(2*mu), 0);

        % update U_L
        U_S = max((mu*C_1+mu*S*V_S'+Y_S*V_S')/(mu*(eye(k)+V_S*V_S')),0);

        % update V_L
        V_S = max(inv(mu*(eye(k)+U_S'*U_S))*(mu*C_2+mu*U_S'*S+U_S'*Y_S)
            ,0);

        % update Y_D, Y_S, Y_U_S, Y_V_S
        Y_D = Y_D+mu*(D-L-S);
        Y_S = Y_S+mu*(S-U_S*V_S);

        obj_D(iter) = norm(D-L-S,'fro');
        error_D(iter) = obj_D(iter)/norm(D,'fro');
        obj_S(iter) = norm(S-U_S*V_S,'fro');
        error_S(iter) = obj_S(iter)/norm(S,'fro');

        % break if it meets the stopping criteria
        if iter > 2 && error_D(iter) < rho_D && error_S(iter) < rho_S &&
            numel(find(S<0)) == 0
                break;
        end
    end
    rank_L(k) = rank(L);
    sparsity_S(k) = nnz(S)/numel(S);
    error_D_final(k) = error_D(iter);
    error_S_final(k) = error_S(iter);

    % store the sign information of low-rank term
    count_L = numel(find(L<0));
    count_L_final(k) = count_L;

    % store the sign information of sparse term
    count_S = numel(find(S<0));
    count_S_final(k) = count_S;

    % store the sign information of U_S term
%     U_S_sign = zeros(size(U_S));
%     U_S_sign = sign(U_S);
%     count_U_S = 0;
%     for i = 1:1:n_1
%         for j = 1:1:k
%             if U_S_sign(i,j) == -1
%                 count_U_S = count_U_S+1;
%                 break;
%             end
%         end
%             if count_U_S == 1
%                 break;
%
%             end
%     end
%     count_U_S_final(k) = count_U_S;

    % store the sign information of V_S term
%     V_S_sign = zeros(size(V_S));
%     V_S_sign = sign(V_S);
%     count_V_S = 0;
%     for i = 1:1:k
%         for j = 1:1:n_2
%             if V_S_sign(i,j) == -1
%                 count_V_S = count_V_S+1;
%                 break;
%             end
%         end
%             if count_V_S == 1
%                 break;
%
%             end
%     end
%     count_V_S_final(k) = count_V_S;
```

```
end


% Parameter initialization
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [lambda,mu,max_iter,rho_D,rho_S,L0,S0,Y_D] = parseInputs(D,opts)

if ~exist('opts','var')
    opts = struct();
end

% parameters
lambda = parseFieldr(opts,'lambda');
mu = parseFieldr(opts,'mu');
rho_D = parseFieldr(opts,'rho_D');
rho_S = parseFieldr(opts,'rho_S');
max_iter = parseFieldr(opts,'max_iter');
L0 = parseField(opts,'L0',rand(size(D)));
S0 = parseField(opts,'S0',rand(size(D)));
Y_D = parseField(opts,'Y_D',rand(size(D)));


% Struct field
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = parseField(stats,field,default)
if isfield(stats,field)
    val = stats.(field);
else
    val = default;
end


% Required field
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = parseFieldr(stats,field)
if isfield(stats,field)
    val = stats.(field);
else
    error('Required field %s not provided',field);
end
```

A-1-3 Proposed algorithm 2

```
function [L,S,count_L_final,count_S_final,count_U_S_final,count_V_S_final
    ,rank_L,sparsity_S,error_D_final,k,iter] = NSRPCA_no_S(D,opts)
%  Proposed algorithm 2: Nonnegative Sparse Component RPCA 2
%  cost function L(L,U_S,V_S) =
%  ||L||_*+<Y_D,D-L-U_S*V_S>+\mu/2||D-L-U_S*V_S||_F^2+...
%  +\mu/2||U_S-C_1||_F^2+\mu/2||V_S-C_2||_F^2
%  constraints: D = L+S
%               U_S>=0, V_S>=0

%   input: D is the image matrix
%          opts:
%          opts.lambda is the coefficient of sparse component
%          opts.mu is the coefficient of error term in the cost function
%          opts.max_iter is the maximum iteration number
%          opts.rho_D is one of the thresholds in the stopping criterion
%          opts.L0 is the initial low-rank component
%          opts.Y_D is the initial Lagrange multiplier

[lambda,mu,max_iter,rho_D,rho_S,L0,S0,Y_D] = parseInputs(D,opts);
for k = 2
    L = L0;

    n_1 = size(D,1);
    n_2 = size(D,2);

    U_S = rand(size(L,1),k);
    V_S = rand(k,size(L,2));
    C_1 = 0;
    C_2 = 0;

    for iter = 1:max_iter
        % update L
        [U,Sigma,V] = svd(D+Y_D/mu-U_S*V_S,'econ');
        Sigma1 = sign(Sigma) .* max(abs(Sigma) - 1/mu, 0);
        L = U*Sigma1*V';
        %rank_L(iter) = rank(L);

        % update U_L
        U_S = max((D*V_S'+Y_D*V_S'/mu-L*V_S'+C_1-sign(U_S*V_S)*V_S'*
            lambda/mu)/(eye(k)+V_S*V_S'),0);

        % update V_L
        V_S = max(inv(eye(k)+U_S'*U_S)*(U_S'*D+U_S'*Y_D/mu-U_S'*L+C_2-U_S
            '*sign(U_S*V_S)*lambda/mu),0);

        % recover S
        S = U_S*V_S;

        % update Y_D, Y_S, Y_U_S, Y_V_S
        Y_D = Y_D+mu*(D-L-S);

        obj_D(iter) = norm(D-L-S,'fro');
        error_D(iter) = obj_D(iter)/norm(D,'fro');

        % break if it meets the stopping criteria
        if iter > 2 && error_D(iter) < rho_D && numel(find(S<0)) == 0
                break;
        end
    end

    rank_L(k) = rank(L);
    sparsity_S(k) = nnz(S)/numel(S);
    error_D_final(k) = error_D(iter);

    % store the sign information of low-rank term
    count_L = numel(find(L<0));
    count_L_final(k) = count_L;

    % store the sign information of sparse term
    count_S = numel(find(S<0));
    count_S_final(k) = count_S;

    count_U_S = numel(find(U_S<0));
    count_U_S_final(k) = count_U_S;

    count_V_S = numel(find(V_S<0));
    count_V_S_final(k) = count_V_S;

end

% Parameter initialization
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [lambda,mu,max_iter,rho_D,rho_S,L0,S0,Y_D] = parseInputs(D,opts)

if ~exist('opts','var')
    opts = struct();
end

% parameters
lambda = parseFieldr(opts,'lambda');
mu = parseFieldr(opts,'mu');
rho_D = parseFieldr(opts,'rho_D');
rho_S = parseFieldr(opts,'rho_S');
max_iter = parseFieldr(opts,'max_iter');
L0 = parseField(opts,'L0',rand(size(D)));
S0 = parseField(opts,'S0',rand(size(D)));
Y_D = parseField(opts,'Y_D',rand(size(D)));


% Struct field
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function val = parseField(stats,field,default)
if isfield(stats,field)
    val = stats.(field);
else
    val = default;
```

```
end


% Required field
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = parseFieldr(stats,field)
if isfield(stats,field)
    val = stats.(field);
else
    error('Required field %s not provided',field);
end
```

A-1-4 IALM algorithm

```
function [L,S,sparsity_S,iter] = basic_RPCA(D,opts)
%  RPCA algorithm
%  cost function L(L,S) = ||L||_*+\lambda||S||_1+<Y,D-L-S>+\mu/2||D-L-S||_F^2
%  constraints: D = L+S

%   input: D is the image matrix
%          opts:
%          opts.lambda is the coefficient of sparse component
%          opts.mu is the coefficient of error term in the cost function
%          opts.max_iter is the maximum iteration number
%          opts.rho is the threshold in the stopping criterion
%          opts.L0 is the initial low-rank component
%          opts.S0 is the initial sparse component

[lambda,mu,max_iter,rho,L0,S0,Y] = parseInputs(D,opts);
L = L0;
S = S0;
n_1 = size(D,1);
n_2 = size(D,2);

for iter = 1:max_iter
        % update L
        [U,Sigma,V] = svd(D-S+Y/mu,'econ');
        Sigma1 = sign(Sigma) .* max(abs(Sigma) - 1/mu, 0);
        L = U*Sigma1*V';

        % update S
        S = sign(D-L+Y/mu) .* max(abs(D-L+Y/mu) - lambda/mu, 0);

        % update Y
        Y = Y+mu*(D-L-S);
        obj(iter) = norm(D-L-S,'fro');
        error = obj(iter)/norm(D,'fro');

        % break if it meets the stopping criteria
        if iter > 2 && obj(iter) < rho*norm(D,'fro')
```

```
                break;
        end
end
    rank_L = rank(L);
    sparsity_S = nnz(S)/numel(S);
    error_final = error;

    % store the sign information of low-rank term
    count_L = numel(find(L<0));

    % store the sign information of sparse term
    count_S = numel(find(S<0));

%     % store the sign information of low-rank term
%     L_sign = zeros(size(L));
%     L_sign = sign(L);
%     count_L = 0;
%     for i = 1:1:n_1
%         for j = 1:1:n_2
%             if L_sign(i,j) == -1
%                 count_L = count_L+1;
%                 break;
%             end
%         end
%             if count_L == 1
%                 break;
%
%             end
%     end
%
%     % store the sign information of sparse term
%     S_sign = zeros(size(S));
%     S_sign = sign(S);
%     count_S = 0;
%     for i = 1:1:n_1
%         for j = 1:1:n_2
%             if L_sign(i,j) == -1
%                 count_S = count_S+1;
%                 break;
%             end
%         end
%             if count_S == 1
%                 break;
%
%             end
%     end


% Parameter initialization
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [lambda,mu,max_iter,rho,L0,S0,Y] = parseInputs(D,opts)
```

```

if ~exist('opts','var')
    opts = struct();
end

% parameters
lambda = parseFieldr(opts,'lambda');
mu = parseFieldr(opts,'mu');
rho = parseFieldr(opts,'rho');
max_iter = parseFieldr(opts,'max_iter');
L0 = parseField(opts,'L0',zeros(size(D)));
S0 = parseField(opts,'S0',zeros(size(D)));
Y = parseField(opts,'Y',zeros(size(D)));



% Struct field
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = parseField(stats,field,default)
if isfield(stats,field)
    val = stats.(field);
else
    val = default;
end


% Required field
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function val = parseFieldr(stats,field)
if isfield(stats,field)
    val = stats.(field);
else
    error('Required field %s not provided',field);
end
```

A-2 Functions

A-2-1 F-measure

```
function [p, r, score] = F_measure_multi(S,GT)

S_reshape = reshape(S(:),[],1);
S_reshape(S_reshape>0) = 255;
GT_reshape = reshape(GT(:,:,1:10),[],1);

S_compare = S_reshape-GT_reshape;
FP = numel(find(S_compare == 255));
FN = numel(find(S_compare == -255));
```

```
TP = numel(find(GT_reshape == 255 )) - FN;
p = TP/(TP+FP);
r = TP/(TP+FN);
score = 2*p*r/(p+r);

end
```

```
function [p, r, score] = F_measure_single(S,GT,k)

S_reshape = S(:,k);
S_reshape(S_reshape>0) = 255;
GT_reshape = reshape(GT(:,:,k),[],1);

S_compare = S_reshape-GT_reshape;
FP = numel(find(S_compare == 255));
FN = numel(find(S_compare == -255));
TP = numel(find(GT_reshape == 255 )) - FN;
p = TP/(TP+FP);
r = TP/(TP+FN);
score = 2*p*r/(p+r);

end
```

A-2-2 Peak Signal-to-noise Ratio (PSNR)

```
function [MSE, ratio] = PSNR_multi(S,GT)
% Calculate peak signal-to-noise ratio between recovered S with ground
% truth data
S_reshape = reshape(S(:),[],1);
GT_reshape = reshape(GT(:,:,1:10),[],1);
MSE = sum((S_reshape - GT_reshape).^2)/(size(S_reshape,1)*size(S_reshape
    ,2));
MAX = 255;
ratio = 10*log10(MAX^2/MSE);

end
```

```
function [MSE,ratio] = PSNR_single(S,GT,k)
% Calculate peak signal-to-noise ratio between recovered S with ground
% truth data

S_reshape = reshape(S(:,k),size(GT(:,:,k),1),[]);
GT_reshape = GT(:,:,k);
MSE = sum((S_reshape - GT_reshape).^2,'all')/(size(S_reshape,1)*size(
    S_reshape,2));
MAX = 255;
ratio = 10*log10(MAX^2/MSE);

end
```

A-2-3 Receiver Operating Characteristic (ROC)

```
function [X,Y,AUC] = ROC_curve_multi(S,GT)
% ROC curve of the video frames

S_reshape = reshape(S(:),[],1);
GT_reshape = reshape(GT(:,:,1:10),[],1);

% range = [0,1]
S_reshape = (S_reshape - min(S_reshape(:))) ./ (max(S_reshape(:)) - min(
    S_reshape(:)));
GT_reshape = (GT_reshape - min(GT_reshape(:))) ./ (max(GT_reshape(:)) -
    min(GT_reshape(:)));
[X,Y,T,AUC] = perfcurve(GT_reshape',S_reshape',1);

end
```

```
function [X,Y,AUC] = ROC_curve_single(S,GT,k)
% ROC curve of the video frames

S_reshape = S(:,k);
GT_reshape = reshape(GT(:,:,k),[],1);

% range = [0,1]
S_reshape = (S_reshape - min(S_reshape(:))) ./ (max(S_reshape(:)) - min(
    S_reshape(:)));
GT_reshape = (GT_reshape - min(GT_reshape(:))) ./ (max(GT_reshape(:)) -
    min(GT_reshape(:)));
[X,Y,T,AUC] = perfcurve(GT_reshape',S_reshape',1);

end
```

A-2-4 Colormap

```
function [TP,FP,TN,FN,S_mask_color] = colormap(S,GT,k)

S_mask = S(:,k);
S_mask(S_mask>0) = 255;
GT_reshape = reshape(GT(:,:,k),[],1);
TP = 0;
FP = 0;
TN = 0;
FN = 0;
S_mask_color = cat(3, S_mask, S_mask, S_mask);

for i=1:size(S,1)
  if(S_mask(i)==255 && GT_reshape(i,1)==255)
      TP=TP+1;
      S_mask_color(i, 1, 2) = 255;
      S_mask_color(i, 1, [1 3]) = 0;
  elseif(S_mask(i)==0 && GT_reshape(i,1)==255)
      FN=FN+1;
      S_mask_color(i, 1, 3) = 255;
      S_mask_color(i, 1, 1:2) = 0;
  elseif(S_mask(i)==0 && GT_reshape(i,1)==0)
```

```
      TN=TN+1;
  else
      FP=FP+1;
      S_mask_color(i, 1, 1) = 255;
      S_mask_color(i, 1, 2:3) = 0;
  end
end

S_mask_color = reshape(S_mask_color,size(GT,1),size(GT,2),[]);

end
```

Appendix B

Theorem

B-1 Karush–Kuhn–Tucker (KKT) conditions

Consider a nonlinear optimization problem with equality and inequality constraints as

$$\begin{gathered}\text{optimize } f(x)\\ \text{s.t. } g_i(x) \leq 0,\\ h_i(x) = 0,\end{gathered} \tag{B-1}$$

where $x \in R^n$ and the number of inequality and equality constraints are denoted by m and k respectively. Then the cost function of the optimization problem can be formulated as

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu^T g(x) + \lambda^T h(x), \tag{B-2}$$

where g(x) = $(g_1(x), \ldots, g_m(x))^T$ and h(x) = $(h_1(x), \ldots, h_k(x))^T$. Scalars λ and μ are the Lagrange multipliers that incorporate the constraints into the cost function. According to [22], the first order KKT conditions are

$$\begin{gathered}\text{for maximizing } f(x)\ \nabla_x \mathcal{L} = \nabla f(x^*) - \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) - \sum_{j=1}^{k} \lambda_j \nabla h_j(x^*) = 0,\\ \text{for minimizing } f(x)\ \nabla_x \mathcal{L} = \nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{k} \lambda_j \nabla h_j(x^*) = 0,\\ \nabla_\mu \mathcal{L} = g_i(x^*) \leq 0 \text{ for } i = 1, \ldots, m,\\ \nabla_\lambda \mathcal{L} = h_j(x^*) = 0 \text{ for } j = 1, \ldots, h,\\ \mu_i \geq 0 \text{ for } i = 1, \ldots, m,\\ \mu_i g_i(x^*) = 0 \text{ for } i = 1, \ldots, m.\end{gathered} \tag{B-3}$$

Bibliography

[1] K. Kiruba, P. Sathiya, and P. AnandhaKumar, "Modified rpca with hessian matrix for object detection in video surveillance on highways," in *2014 Sixth International Conference on Advanced Computing (ICoAC)*, pp. 242–247, IEEE, 2014.

[2] A. Baghaie, R. M. D'souza, and Z. Yu, "Sparse and low rank decomposition based batch image alignment for speckle reduction of retinal oct images," in *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pp. 226–230, IEEE, 2015.

[3] A. Sobral, T. Bouwmans, and E.-h. ZahZah, "Double-constrained rpca based on saliency maps for foreground detection in automated maritime surveillance," in *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, IEEE, 2015.

[4] Z. Gao, L.-F. Cheong, and Y.-X. Wang, "Block-sparse rpca for salient motion detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 10, pp. 1975–1987, 2014.

[5] B.-H. Chen, L.-F. Shi, and X. Ke, "A robust moving object detection in multi-scenario big data for video surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 982–995, 2018.

[6] Y. Dai, Y. Wu, Y. Song, and J. Guo, "Non-negative infrared patch-image model: Robust target-background separation via partial sum minimization of singular values," *Infrared Physics & Technology*, vol. 81, pp. 182–194, 2017.

[7] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[8] I. T. Jolliffe, "Principal components in regression analysis," in *Principal component analysis*, pp. 129–155, Springer, 1986.

[9] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.

[10] M. Shakeri and H. Zhang, "Real-time bird detection based on background subtraction," in *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pp. 4507–4510, IEEE, 2012.

[11] P.-H. Lee, C.-C. Chan, S.-L. Huang, A. Chen, and H. H. Chen, "Extracting blood vessels from full-field oct data of human skin by short-time rpca," *IEEE transactions on medical imaging*, vol. 37, no. 8, pp. 1899–1909, 2018.

[12] H. Ji, S. Huang, Z. Shen, and Y. Xu, "Robust video restoration by joint sparse and low rank matrix approximation," *SIAM Journal on Imaging Sciences*, vol. 4, no. 4, pp. 1122–1142, 2011.

[13] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Real-time online singing voice separation from monaural recordings using robust low-rank modeling.," in *ISMIR*, pp. 67–72, Citeseer, 2012.

[14] T. Kumar and K. Verma, "A theory based on conversion of rgb image to gray image," *International Journal of Computer Applications*, vol. 7, no. 2, pp. 7–10, 2010.

[15] G. Hinton and A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical report, University of Toronto*, 2009.

[16] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[17] M. W. Berry and M. Browne, "Email surveillance using non-negative matrix factorization," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 249–264, 2005.

[18] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2012.

[19] Y. Wang, Y. Jia, C. Hu, and M. Turk, "Non-negative matrix factorization framework for face recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 19, no. 04, pp. 495–511, 2005.

[20] A. B. Hamza and D. J. Brady, "Reconstruction of reflectance spectra using robust nonnegative matrix factorization," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, pp. 3637–3642, 2006.

[21] V. P. Pauca, J. Piper, and R. J. Plemmons, "Nonnegative matrix factorization for spectral data analysis," *Linear algebra and its applications*, vol. 416, no. 1, pp. 29–47, 2006.

[22] I. Griva, S. G. Nash, and A. Sofer, *Linear and nonlinear optimization*, vol. 108. Siam, 2009.

[23] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562, 2001.

[24] X. Peng, C. Lu, Z. Yi, and H. Tang, "Connections between nuclear-norm and frobenius-norm-based representations," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 1, pp. 218–224, 2016.

[25] T.-H. Oh, Y.-W. Tai, J.-C. Bazin, H. Kim, and I. S. Kweon, "Partial sum minimization of singular values in robust pca: Algorithm and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 744–758, 2015.

[26] X. Yuan and J. Yang, "Sparse and low-rank matrix decomposition via alternating direction methods," *preprint*, vol. 12, no. 2, 2009.

[27] T. Bouwmans and E. H. Zahzah, "Robust pca via principal component pursuit: A review for a comparative evaluation in video surveillance," *Computer Vision and Image Understanding*, vol. 122, pp. 22–34, 2014.

[28] P. Rodriguez and B. Wohlberg, "Fast principal component pursuit via alternating minimization," in *2013 IEEE International Conference on Image Processing*, pp. 69–73, IEEE, 2013.

[29] X. Cao, L. Yang, and X. Guo, "Total variation regularized rpca for irregularly moving object detection under dynamic background," *IEEE transactions on cybernetics*, vol. 46, no. 4, pp. 1014–1027, 2015.

[30] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *arXiv preprint arXiv:1009.5055*, 2010.

[31] X. Luan, B. Fang, L. Liu, W. Yang, and J. Qian, "Extracting sparse error of robust pca for face recognition in the presence of varying illumination and occlusion," *Pattern Recognition*, vol. 47, no. 2, pp. 495–508, 2014.

[32] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[33] V. Mahadevan and N. Vasconcelos, "Spatiotemporal saliency in dynamic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 171–177, 2010.

[34] D. Salomon, *Data compression: the complete reference.* Springer Science & Business Media, 2004.

[35] X. Zhou, C. Yang, and W. Yu, "Moving object detection by detecting contiguous outliers in the low-rank representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 3, pp. 597–610, 2012.

[36] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1568–1575, IEEE, 2012.

[37] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[38] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

Glossary

List of Acronyms

PCA	Principal Component Analysis
PCP	Principal Component Pursuit
ALM	Augmented Lagrange Multiplier Method
ADM	Alternating Direction Method
RPCA	Robust Principal Component Analysis
APG	Accelerated Proximal Gradient
OCT	Optical Coherence Tomography
EALM	Exact ALM
IALM	Inexact ALM
KKT	Karush–Kuhn–Tucker
NMF	Nonnegative Matrix Factorization
CNMF	Constrained NMF
SNMF	Structured NMF
SVD	Singular Value Decomposition
FPCP	Fast PCP
TVRPCA	Total Variation Regularized RPCA
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
MSE	Mean Squared Error
PSNR	Peak Signal-to-noise Ratio
ROC	Receiver Operating Characteristic
TPR	True Positive Rate

FPR	False Positive Rate
AUC	Area Under Curve