COROLA: A Sequential Solution to Moving Object Detection Using Low-rank Approximation

Moein Shakeri, Hong Zhang
Department of Computing Science
University of Alberta, Edmonton, Alberta, CANADA
shakeri,hzhang@ualberta.ca

Abstract

Extracting moving objects from a video sequence and estimating the background of each individual image are fundamental issues in many practical applications such as visual surveillance, intelligent vehicle navigation, and traffic monitoring. Recently, some methods have been proposed to detect moving objects in a video via low-rank approximation and sparse outliers where the background is modeled with the computed low-rank component of the video and the foreground objects are detected as the sparse outliers in the low-rank approximation. All of these existing methods work in a batch manner, preventing them from being applied in real time and long duration tasks. In this paper, we present an online sequential framework, namely contiguous outliers representation via online low-rank approximation (COROLA), to detect moving objects and learn the background model at the same time. We also show that our model can detect moving objects with a moving camera. Our experimental evaluation uses simulated data and real public datasets and demonstrates the superior performance of COROLA in terms of both accuracy and execution time.

Keywords: Moving Object Detection, Online Low Rank Approximation, Markov Random Fields, Online Background modeling
1. Introduction

Moving object detection and background estimation are fundamental in various applications of computer vision and robotics such as visual surveillance [1], traffic monitoring [2], vehicle tracking and navigation [3], and avian protection [4]. Many methods have been proposed to extract objects from a sequence of images with a stationary camera [5, 6] or with a moving camera [7, 8, 9]. These methods can be grouped into different categories. Motion-based methods [10, 11] are one category where motion information of the image pixels is used to separate the foreground from the background. These methods work based on the assumption that foreground objects move differently from the background. Therefore it is possible for these methods to classify pixels according to their movement characteristics even in the case of significant camera motion. However, these methods require a point tracking algorithm to identify the foreground, which can be difficult especially with large camera motion [12]. Also, they are limited in dealing with dynamic background or noisy data [13].

Another popular category for moving object detection methods is background subtraction [14], which compares the pixels of an image with a background model and considers those that differ from the background model as moving objects. Thus, building a background model plays a critical role in background subtraction methods. Some conventional algorithms for background modelling are single Gaussian distribution [15], Gaussian mixture model [16], and kernel density estimation [17]. These methods model the background for each pixel independently and so they are not robust against global variations such as illumination changes. Recently, a new approach to background modelling, namely low-rank matrix approximation, has been developed to tackle the problem of background modeling [18]. This approach is based on the observation that the background model in an image sequence can be defined by those pixels that are temporally linearly correlated [19]. By capturing the correlation between images it can naturally handle global variations. Algebraically speaking, if an image is vectorized in a column and all images are concatenated into a 2D
matrix, then the columns are dependent and its low-rank matrix represents the background model of the images. As a result, the background modeling problem is converted to the low-rank approximation problem. In general, by decomposing an input matrix of vectorized images into a low-rank matrix and a sparse matrix, the low-rank and sparse matrices correspond to the background model and the foreground objects in the image sequence respectively. Our COROLA algorithm described in this paper adopts the low-rank approximation approach. We will detail representative algorithms in this approach in Section 2.

Existing background subtraction algorithms based on low-rank approximation operate in a batch manner; i.e., all images whose background model is to be constructed are first collected and then used to build a data matrix whose low-rank approximation is computed. This unfortunately limits the application of the low-rank approximation approach in terms of its efficiency and accuracy. In this paper, we offer an algorithm, COROLA, that performs low-rank approximation in a sequential manner so that its complexity does not grow with the number of images in the sequence. In addition, through image registration, our algorithm is able to handle the case of a moving camera due to the adaptive nature of the background model that is being learned. The main contributions of this paper are as follows.

1. We propose an online formulation of the low-rank approximation algorithm for foreground object detection. The proposed formulation enables online application without requiring an entire image sequence, as in the batch formulation.  
2. COROLA uses a fixed window of images to perform low-rank approximation and so it is useful for continuous operation, which cannot be achieved by the batch formulation due to matrix decomposition and memory storage.
3. In the case of significant camera motion, a batch formulation has the limitation that the first and the last images of a sequence must be similar to find the low-rank matrix. However, in the case of a moving camera, there is in general no similarity between the first and the last images in a sequence. Our proposed COROLA algorithm does not require a stationary background.
The remainder of the paper is organized as follows. Related works on foreground detection via low-rank and sparse decomposition are summarized in Section 2. Section 3 explains the details of COROLA for foreground detection and background estimation, followed by the introduction of our online formulation via greedy bilateral sketch [20]. Experimental results and discussion are presented in Section 4, followed by concluding remarks in Section 5.

2. Foreground Detection via Low Rank and Sparse Decomposition

In recent years, many algorithms have been developed for foreground detection based on low-rank matrix approximation with robust principal component analysis (RPCA) [21]. RPCA decomposes a given matrix $D$ into low-rank matrix $L$ and sparse matrix $S$ called outliers. Different techniques exist for low-rank approximation including augmented Lagrangian multiplier (ALM) [22], linearized alternating direction method with an adaptive penalty (LADMAP) [23], and singular value thresholding (SVT) [24]. All of these techniques need all the data in a batch optimization, in order to compute the low-rank matrix and the sparse outliers. Due to batch processing, the following two problems occur: memory storage and time complexity. In continuous monitoring tasks or video processing, if matrix $D$ is built with a large number of images memory storage will be a problem [25]. In addition, by increasing the size of the input matrix $D$, time complexity for the matrix decomposition is also increasing.

To address the problem of time complexity, some efficient algorithms have been proposed. Zhou et al. proposed a method for RPCA using bilateral random projections (BRP) which is called “Go Decomposition” (GoDec) [26]. Semi-Soft GoDec (SSGoDec) and Greedy SSGoDec methods [20] are extensions of GoDec to accelerate it. Although these algorithms reduce the computation time of low-rank approximation, they still are not satisfactory for applications such as visual surveillance and robot navigation due to their batch formulation. In many applications, online processing is critical and batch methods are infeasible. One of the best known batch processing algorithms is the “detecting contiguous out-
liers in the low-rank representation” (DECOLOR) method [27]. This method uses a priori knowledge of the foreground objects that they should be connected components of relatively small size. Using this constraint in the method, DECOLOR provides promising results; however, due to batch processing, it still suffers from memory storage and time complexity problems. Furthermore, in the case of a moving camera, the current image is no longer similar to the first images in matrix $D$, and therefore this kind of low-rank approximation methods cannot detect foreground appropriately. In general, batch processing methods cannot operate on a continuous basis and cannot deal with a moving camera. Although, DECOLOR has introduced an implementation for moving camera, it only works for short video sequences with small camera motion.

To overcome the limitations of batch processing methods, Feng et al. [28] proposed an online robust principal component analysis via stochastic optimization (OR-PCA) and Javed et al. [29] used this technique for online foreground detection. Their method first extracts outliers from each image using OR-PCA and then uses Markov Random Field (MRF) to improve the quality of foreground segmentation. However, they did not solve the problem of foreground detection within a unified single optimization framework, i.e., MRF is only applied once to improve the outliers of OR-PCA and basically there is no alternating learning algorithm to learn from the MRF outputs to update the OR-PCA, thus the reported performance is not competitive with respect to the literature.

In this paper, we introduce a novel non-convex closed-form formulation for detection of moving objects named (COROLA). It solves the challenges mentioned above in memory storage and time complexity. COROLA is also able to extract moving objects using a moving camera on a continuous basis, which cannot be achieved in general by a batch processing method especially in the case of large camera motion.
3. Online Moving Object Detection by COROLA

In this section, we focus on online detection of moving objects for both static and moving cameras. We first formulate the problem of background modelling and foreground object detection and then our COROLA algorithm, which computes the low-rank approximation sequentially, is described in details.

3.1. Notations and Formulation

Let $X_j \in \mathbb{R}^m$ be the $j^{th}$ image in a sequence, expressed as a column vector of $m$ pixels. Then, $D = [X_1, ..., X_n] \in \mathbb{R}^{m \times n}$ is a matrix of $n$ images and the $i^{th}$ pixel in the $j^{th}$ image is denoted as $x_{ij}$. To indicate foreground for an observed image $j$, we use a binary indicator vector $S = [s_1, s_2, ..., s_m]^T$ as the foreground support where

$$s_i = \begin{cases} 
0 & \text{if } i \text{ is background} \\
1 & \text{if } i \text{ is foreground}
\end{cases} \quad (1)$$

Also, we use the function $P_S(X) \in \mathbb{R}^{|S|_0}$ to construct a vector of at most $m$ foreground pixels of image $X$. Note that $L_0$-norm $|S|_0$ equals the number of non-zero elements in $S$. In a matrix with more than one column, $P_S$ constructs multiple columns each by applying $P_S$ to a column in the input matrix.

With the above notations, the problem of background modelling and foreground object detection via sequential low-rank approximation and contiguous outlier representation solves the following optimization problem.

$$\min_{U,V,S} \frac{1}{2} \|P_S(X - UV)\|_F^2 + \beta_1 \|S\|_1 + \gamma \|\Phi(S)\|_1$$

$$\text{s.t. } \text{rank}(U) \leq r, \|V\|_0 \leq r \quad (2)$$

where $X \in \mathbb{R}^m$ is an observed image, $r$ is upper bound on the rank of the basis matrix $U \in \mathbb{R}^{m \times r}$, and $V \in \mathbb{R}^r$ is a coefficient vector. $\Phi(S)$ means the difference between neighboring pixels and it is computed by $\|\Phi(S)\|_1 = \sum_{(i,k) \in \mathcal{E}} |s_i - s_k|$ and $\mathcal{E}$ is the neighborhood clique. The objective function defined in (2) is non-convex and involves both continuous and discrete variables. The first term tries to compute the low-rank representation of input image $X$ by first expressing it
as a linear combination of the background image $U$ and then penalizing only the foreground pixels using extraction function $P_S$. The second and the third terms of (2) find continuous and small outliers to represent foreground mask. Specifically, the second term imposes a sparsity constraint on the foreground mask $S$; i.e., the foreground pixels should be few. The third term imposes a connectivity constraint on mask $S$ to account for correlation between neighboring pixels of an image. By minimizing (2) we can estimate the best low-rank representation of an input image and detect foreground objects, accurately. However, solving this joint optimization in one step is difficult. Therefore, people use a two-step alternating optimization procedure by separating it to low-rank approximation involving $U$ and $V$ and contiguous sparse optimization involving $S$ to obtain background estimation and foreground detection, respectively. In the first step people treat (2) as minimization over $U$ and $V$, for which we introduce an online approach via greedy semi-soft GoDec (Gre-SSGoDec) method rather than the SOFT-IMPUTE algorithm [30] in batch methods. In the second step, minimization over $S$ is conducted. In addition, we use the combination of Gaussian Mixture Model (GMM) and first order MRF with binary labels in the second step.

3.2. Online Low-Rank Approximation

In this section, we describe our sequential method to compute the low rank background model of an image sequence and the foreground as its sparse outliers, in a way that is suitable for continuous and real time operation. Let $L = UV$ be a low rank structure of the data matrix $D$ defined in the previous section. In a batch method, matrix $D$ is factorized to compute $L$. Factorization is computationally expensive for a large $D$, especially for continuous estimation when the number of images in $D$ increases. In our sequential formulation, we adopt an online updating approach instead for optimization over $U$ and $V$, (2)
can be rewritten as:
\[
\min_{U,V} \frac{1}{2} \|P_S(X - UV)\|_F^2
\]
\[\text{s.t. } \text{rank}(U) \leq r, \|V\|_0 \leq r \]  

\( (3) \)

Initialization Step: With a small number of first those images at the beginning of a sequence no fewer than the rank of the background model, we initialize \( U \) and \( V \) with a batch method. Since this step is performed only once, the complexity of using a batch formulation is not an issue. Now to solve \( (3) \) we use the following three steps, repeatedly on the remainder of the image sequence or indefinitely on a video feed.

Step 1: Because every two consecutive images in a sequence are similar, we can update coefficient vector \( V \) (or \( U \)) for the current image via background model \( U \) (or \( V \)) computed for the previous image. To update \( V \) with the previous \( U \), \( (3) \) can be changed to:
\[
\hat{V}_j = \arg\min_{V} \frac{1}{2} \|P_S(X_j - U_{j-1}V)\|_F^2 + \beta_1 \|V\|_2
\]
where \( X_j \in \mathbb{R}^m \) is the current image. By fixing \( U_{j-1} \), \( (4) \) is a least squares problem and can be solved by
\[
\hat{V}_j = (U_{j-1}^TU_{j-1})^\dagger U_{j-1}^TX_j
\]
where \((,)^\dagger\) is the Moore Penrose pseudoinverse \[20\].

Step 2: We use the same approach as in \[28\] by defining two auxiliary parameters \( A \in \mathbb{R}^{r \times r} \) and \( B \in \mathbb{R}^{m \times r} \) in which to keep the information about \( V \) and \( U \) as follows:
\[
A_j = \bar{A} + \hat{V}_j\hat{V}_j^T
\]
\[
B_j = \mathcal{P}_{S_i}(\bar{B}) + \mathcal{P}_{S_i}(U_{j-1})\hat{V}_j\hat{V}_j^T
\]
where \( \bar{A} = A_{j-1} \) and \( \bar{B} = B_{j-1} \) for the first iteration or \( \bar{A} = A_j \) and \( \bar{B} = B_j \) for further iterations. In contrast with \[28\] we update \( B \), only for those pixels that have foreground support \( s_i = 1 \). Therefore, the number of rows in \( B \) is variable and equal to \(|S|_0\) in each iteration. In this step additive \( A \) and \( B \) save
all previous information of $U$ and $V$ and are updated for the current image. By increasing the values of $A$ and $B$, the obtained background model becomes stable.

**Step 3:** Based on the obtained $V_j$, $A_j$ and $B_j$, we update $U_j$ for the current image as follows [28]:

$$\hat{U}_j = \arg\min_U Tr[U(A_j + \beta_i I)U^T] - Tr(U^TB_j)$$ (7)

where $U = P_{S,i}(U_{j-1})$ means we update $\hat{U}_j$ for those pixels that have foreground mask $s_i = 1$. For the first iteration, $s_i = 1$ for all pixels of the current image and so the number of rows in $B_j$ and $\hat{U}_j$ is the same as that in the input image; subsequently, the number of rows decreases in succeeding iterations as the foreground area is decreased. (7) can be solved with a simple iterative algorithm presented in [28]. Since COROLA is an iterative algorithm based on (2) and the size of $\hat{U}_j$ and $B_j$ is changed in each iteration, in this implementation we keep $\hat{U}_j$, $\hat{V}_j$, $A_j$, and $B_j$ that show their values after the first iteration. We keep these variables to be consistent with the first iteration of the next input image. Also these variables have the most information to build the background model for the current image, which is computed by $L_j = \hat{U}_j \hat{V}_j$. However, foreground detection depends on the obtained $S$ from the second step of solving (2), and the algorithm continues to iterate until the convergence criteria are met. Because for dynamic backgrounds, outliers are combination of the foreground object and moving parts of the background as noise (i.e. see waves). These moving parts do not affects on background model, but they create many false positive in the foreground mask $S$. We will explain the convergence criteria after solving the second step of (2).

### 3.3. Online Foreground Detection

Now we investigate how to compute foreground mask $S$ given the obtained residual $E_j = X_j - L_j$ ($L_j$ is computed in background modeling in the previous section). The goal now is to find indicator vector $S$ on $E_j$. Assuming that the foreground objects are relatively small connected components, we can model
the foreground mask $S$ by a Markov Random Field (MRF) [31]. Let graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices that correspond to the pixels of an image and $\mathcal{E}$ is the set of edges that connect neighboring pixels. Then, by defining an energy function of $S$

$$
\sum_{i \in \mathcal{V}} \beta_2(s_i) + \sum_{(i,k) \in \mathcal{E}} \gamma_{i,k}|s_i - s_k| \quad (8)
$$

which is called “Icing model” in the literature, and an example of MRF, we can derive the foreground mark $S$. The first and the second terms impose sparsity and continuity on $S$, in a way that is similar to the second and the third terms of (2) and shows that $S$ can be modeled using MRF [31]. However, extracting foreground objects from $E$, which is combination of outliers and noise, would not be accurate especially in noisy environment like dynamic backgrounds or when using a moving camera. In most cases we need to separate reliable outliers representing true foreground from noise in estimating foreground support $S$. In our applications, noise comes from a complicated and dynamic background such as waving trees or sea waves and we have to classify them as background.

Here, we describe outliers with a Gaussian model, $\mathcal{N}(\mu, \sigma^2)$. Using this model on the outliers enables us to control the complexity of the background variations and also amplifies the true outliers against noise using (9). In our study, adaptive Gaussian Mixture Model (GMM) [32] is used for each component of $E$ to separate the outliers from noise. As in most of the cases, three Gaussian components are sufficient in modeling $E$ to separate foreground $F$ from noise [32]. Fig. 1 shows the effect of using GMM on $E$ for dynamic backgrounds. The middle figure shows the obtained residual $E$. After obtaining $E$, we normalize it and extract outliers $F$ from noise using Gaussian model (right figure). So, to solve the second step of (2) we prepare $\hat{E}$ with a simple update rule as follows:

$$
\hat{E}_j = \alpha E_j + (1 - \alpha) F_j \quad (9)
$$

where $E_j = X_j - L_j$ and $F_j$ is obtained outliers using GMM on the current image ($j^{th}$ image of the sequence). $\alpha \in [0, 1]$ is a constant that controls the magnitude
Figure 1: The effects of using GMM on outliers obtained from low rank approximation on noisy and dynamic background. The left figure shows an input image, and the middle and right figures show the obtained outliers $E$ and $\hat{E}$, respectively.

of noise so that small $\alpha$ would be used for noisy data (i.e. for moving cameras). In all of our experiments $\alpha = 0.1$.

Now we can solve the second step of our optimization problem that extracts moving objects from outliers, and (2) can be rewritten as the following equation to minimize the energy over $S$ via obtained outliers $\hat{E}$.

$$\min_S \frac{1}{2} ||P_S(\hat{E})||_F^2 + \beta_2 ||S||_1 + \gamma \|\Phi(S)\|_1$$

$$= \sum_{i:s_i=1} \hat{E}_i^2 + \beta_2 \sum_{i} s_i + \gamma \|\Phi(S)\|_1$$

The first term of (10) is constant and therefore (10) is the first order MRF with binary labels (the same as (8)), which can be solved using graph-cut [33], [34]. The result of (10) is the binary mask $S$, which indicates the foreground pixels of $X_j$. So far, the first iteration of (2) is done and based on mask $S$, the next iteration starts from (4). In our experiments, COROLA converges in approximately $r$ iterations where $r$ is the rank of data in the sequence. In Algorithm 1, all steps of COROLA are summarized.

3.4. Online Moving Object Detection with a Moving Camera

In this part, we extend our moving object detection method to the case of a moving camera. As we mentioned in Section 1, due to the dissimilarity between the first and the last images in a sequence, a batch method is not able to deal
Algorithm 1 Online Moving Object Detection by COROLA

1: Initialize: GMM parameters, $\beta_1$, $\beta_2$, $\gamma$, $\alpha_1$, $r$, $A$, and $B$

2: for $j = 1 : n$
3:   Input data: $X_j$
4:   $t = 1$ and $s_i = 1$, $i = \{1, ..., m\}$
5:   repeat
6:     If $t = 1$
7:        $\bar{A} \leftarrow A_{j-1}$, $\bar{B} \leftarrow B_{j-1}$, $\bar{U} \leftarrow U_{j-1}$
8:     else
9:        $\bar{A} \leftarrow A_j$, $\bar{B} \leftarrow B_j$, $\bar{U} \leftarrow U_j$
10:   end If
11:   $\hat{V}_j \leftarrow \text{argmin}_{V} \frac{1}{2} \|P_S(X_j - \bar{U}V)\|_F^2 + \beta_1\|V\|_2$
12:   $A_j \leftarrow \bar{A} + \hat{V}_j\hat{V}_j^T$, $P_{S_i}(B_j) \leftarrow P_{S_i}(B_{j-1}) + P_{S_i}(\bar{U})\hat{V}_j\hat{V}_j^T$
13:   $\hat{U}_j \leftarrow \text{argmin} \text{Tr}[C^T(A_j + \beta_1 I)C] - \text{Tr}(C^T(B_j))$, where $C = P_{S_i}(\bar{U})$
14:   $E_j \leftarrow X_j - L_j$, compute $F_j$ over $E_j$ from [32]
15:   $\hat{E}_j \leftarrow \alpha E_j + (1 - \alpha)F_j$
16:   Compute cost of assigning labels using $\hat{E}_j$ to optimize $S$
17:   $S \leftarrow \text{argmin}_{S} \beta_2 \sum_i s_i + \gamma \|\Phi(S)\|_1$
18:   If $t = 1$,
19:      $1\hat{U}_j \leftarrow \hat{U}_j$, $1\hat{V}_j \leftarrow \hat{V}_j$, $1A_j \leftarrow A_j$, and $1B_j \leftarrow B_j$
20:   end If
21:   If $t \geq r$
22:      break
23:   else
24:      $t \leftarrow t + 1$
25:   end If
26:   until convergence
27: Output: $S_j$, $L_j = 1\hat{U}_j1\hat{V}_j$
28: $\hat{U}_j \leftarrow 1\hat{U}_j$, $\hat{V}_j \leftarrow 1\hat{V}_j$, $A_j \leftarrow 1A_j$, and $B_j \leftarrow 1B_j$
29: end for
with continuous processing using a moving camera. However, in online methods the background model is dynamic and dissimilarity problem does not occur. In our method, we build the background model for the current image and based on a transformation function between the current and the new image, the model is transformed to be matched with the new image. Then we can update it for the new image to detect the foreground objects. Note that the background model is transformed through the time. So the problem of using a moving camera is the transformation of the low-rank structure to the new input image.

To show the capability of our method in comparison with the DECOLOR method, we use the same image transformation parameters as DECOLOR. Let \( \tau_j \) be a transformation that maps \( X_{j-1} \) to \( X_j \). This transformation is obtained from an affine transformation estimated from their 2D images. We also assume \( X_{j-1} = \hat{U}_{j-1}\hat{V}_{j-1} \) and there is no changes into both images except for affine transformation and so \( X_j = \tau \circ X_{j-1} \). For the sake of brevity, we state without proof that the following equation allows us to reconstruct the current view \( X_j \) from the background model and the registration transform \( \tau_j \).

\[
X_j = \tau_j \circ X_{j-1} = (\tau_j \circ \hat{U}_{j-1})\hat{V}_{j-1}
\]

From (11) the transformation only changes \( \hat{U} \). In fact, we just need to transform \( B \) via \( \tau \) only once for the first iteration of each input image where \( \hat{U}_{j-1} = \tau \circ U_{j-1} \) and \( \hat{B}_{j-1} = \tau \circ B_{j-1} \). In (6), \( A \) remains unchanged, because \( \hat{V} \) is independent from \( \tau \). Based on the above assumptions and (11), \( U_j = \hat{U}_{j-1} \) and \( V_j = \hat{V}_{j-1} \). After the transformation, some elements of \( \hat{U}_{j-1} \) and \( \hat{B}_{j-1} \), which are related to the pixels on the border of the current image, have no corresponding pixels and we have to estimate them using other pixels. To solve the problem, first we normalize both \( \hat{U} \) and the current image to \([0,1]\). Then, using \( X_j \) and \( V_{j-1} \) (or \( V_j \)) we estimate missing pixels of \( \hat{U}_{j-1} \) with replacing by the corresponding values obtained from [20] as follows and ensure they lie in the correct range.

\[
\hat{U}_{j-1} = X_j\hat{V}_{j-1}(V_{j-1}\hat{V}_{j-1}^T)^\dagger
\]

For estimating missing pixels of transformed \( B_{j-1} \), we normalize both \( B_{j-1} \)
Figure 2: An example of COROLA for a moving camera. (a) input image from a sequence (b) background model (c) $E$, (d) $\hat{E}$, (e) $S$, and (f) extracted foreground object using mask $S$. Red lines show the processing area.

and $\tilde{U}_{j-1}\hat{V}_j\hat{V}_j^T$ and we replace those missing values of $\tilde{B}_{j-1}$ with $\tilde{U}_{j-1}\hat{V}_j\hat{V}_j^T$ (from (6)) and ensure they lie in the correct range.

Based on the experimental results, this approach can estimate missing pixels of $U$ and $B$ after transformation. In addition, the GMM for the previous $E_{j-1}$ should be transformed via $\tau$ to match with the current $E_j$. After transforming $U, B$, we can apply the COROLA method for a static camera to build the background model and detect the foreground objects. Fig. 2 shows a sample image and its computed background model and extracted moving object via COROLA and also the intermediate results.

The complexity of our sequential low-rank approximation by COROLA consists of contributions from three major steps. The computational complexity of the first step is $O(mr)$. The second and the third steps of the low-rank approximation in our model are $O(r^2 + mr)$ and $O(mr^2)$, respectively. Therefore, the overall complexity of COROLA for the low-rank approximation step is
4. Experimental Results

In this section, we perform two sets of experiments on synthetic data and real datasets and show quantitative and qualitative results. For quantitative evaluation, we use pixel-level precision and recall, which are defined as follows:

\[
\text{precision} = \frac{TP}{TP + FP}, \quad \text{recall} = \frac{TP}{TP + FN}
\]

where TP, FP, TN, and FN are the numbers of true positives, false positives, true negatives and false negatives, in pixels, respectively. Also, instead of using precision-recall curves, we use F-measure to show the overall accuracy.

\[
F\text{-measure} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

4.1. Synthetic Data

In this set of experiments, we use synthetic data to control noise and to show the capability of the COROLA method. The synthesized images are 30 \times 100 pixels \((m = 3000)\). We use \(n = 200\) images.

To visualize the results we show all images in a 2D matrix where each column shows one image of the sequence. We generate the input data \(D\) by adding a foreground to a background matrix \(B\). For generating the foreground and background we use the same approach as DECOLOR method. The background matrix \(B = UV\) is generated via \(U \in \mathbb{R}^{m \times r}\) and \(V \in \mathbb{R}^{r \times n}\) with random samples from a standard normal distribution. An object with a small size is superimposed on each image in matrix \(B\), and shifts from left to right of the images by one pixel per image, until the right border of the image. The motion direction of the object is then reversed, and the process repeats. Fig. 3(b) shows some selected images. The intensity of this object is independently sampled from a uniform distribution. Also, we add i.i.d Gaussian noise \(\epsilon\) to \(D\) with the corresponding signal-to-noise ratio defined as

\[
\text{SNR} = \sqrt{\frac{\text{var}(B)}{\text{var}(\epsilon)}}
\]
Figure 3: An example of synthetic data. (a) shows matrix $B \in \mathbb{R}^{3000 \times 200}$, with $m = 3000$, $n = 200$, and rank $r = 5$, where $B = UV$, $U \in \mathbb{R}^{3000 \times 5}$, and $V \in \mathbb{R}^{5 \times 200}$. (b) shows some sample images from selected column of $B$, where an object is superimposed each of them. The object is represented by a red box in the first image in (b). other images show the movement of the object to left and right of the image, frequently. (c) shows a sample of generated matrix $D$ as the input data.

Figs. 3(a), (b) and (c) show an example of generated $B$, the movement of generated foregrounds and the obtained matrix $D$.

We test the COROLA method and compare it with DECOLOR in terms of different SNR ratios, different ranks of matrix, and different sizes of the foreground object. One sample of our experiments with different SNR ratios between COROLA and DECOLOR methods is shown in Fig. 4. In the first row of Fig. 4, with $SNR = 10$ both COROLA and DECOLOR methods have roughly the same results for extracting the foreground object, but when we increase noise in the second row ($SNR = 1$), COROLA method works better than DECOLOR in extracting the moving object. That is mainly because, using GMM to compute the coefficients of outliers plays an important role to separate the foreground object from noise. Tuning up the outliers coefficient via GMM enables us to separate noise and outliers especially in a noisy environment and the result becomes more and more accurate over the time.

To evaluate COROLA in comparison with DECOLOR, we tested the effects
of some parameters such as SNR, rank of matrix $D$, and size of the object. Fig. demonstrates the effects of these parameters on our method and DECOLOR method. The first row of Fig. illustrates the effect of noise in both COROLA and DECOLOR methods, when we change the SNR ratio from 8 to 1 in different ranks. The columns from left to right show our experiments in different ranks of 1, 3, and 5. The second row of Fig. shows the accuracy of COROLA and DECOLOR to extract the moving object of different sizes. This experiment illustrates the capability of our method is comparable with DECOLOR in terms of average F-measure. Although, the result of DECOLOR method is a little more accurate than COROLA for large objects, by reducing the size of object, COROLA generates a better result than DECOLOR even when we increase the rank of matrix $D$ from 1 to 5.

4.2. Real Data

In this section, we use real datasets to conduct quantitative and qualitative evaluation of COROLA and compare it with DECOLOR, MOG, SSGoDec, and ORPCA+MRF. The real datasets used are popular in moving object detection.
Figure 5: First row: the comparison of F-measure between COROLA and DECOLOR with different signal-to-noise (SNR) ratio. Second row: the comparison of F-measure between COROLA and DECOLOR with different object size and publicly available, and they include ‘Change Detection’ [35], ‘I2R’ [36], and ‘Wallflower’ [37]. Table 1 shows the length and image size of these datasets.

4.2.1. Evaluation by accuracy

Fig. 6 shows the qualitative results of COROLA for background estimation and foreground detection for all sequences of Table 1. Fig. 6 also shows the role

Table 1: Details of all sequences used in our experiments for stationary camera

| Dataset         | Sequences          | Size × #frames     |
|-----------------|--------------------|--------------------|
| Change Detection| Canoe              | [320,240] × 1189   |
|                 | Overpass           | [320,240] × 3000   |
| I2R             | Water surface      | [160,128] × 48     |
|                 | Fountain           | [160,128] × 523    |
|                 | Curtain            | [160,128] × 2964   |
|                 | Hall               | [176,144] × 1927   |
| Wallflower      | Waving trees       | [160,120] × 287    |
of GMM to separate outliers from noise. These results are shown in columns (d) and (e) as $E$, and $\hat{E}$, respectively. The results in Fig. 6 demonstrate the capability of COROLA to detect moving objects and background modeling accurately. The estimated background in the first row has some ghost because the input image is the 23rd of the sequence and the parameters have not been learned well enough to build an accurate background. In general, for the short sequences the computed background model by DECOLOR method is more accurate than COROLA because online methods need some samples for training to be stable. However, for the long sequences COROLA can provide comparable results with DECOLOR.

We also compare COROLA quantitatively with other methods. Table 2 compares COROLA with MOG, SSGoDec, and DECOLOR in terms of F-measure. In most of the cases COROLA is comparable with DECOLOR; however, for three sequences Fountain, Canoe and Overpass COROLA works much better than DECOLOR. Because in these sequences the objects move very slowly (i.e. Canoe) or stop for a long time (i.e. Overpass and Fountain) and DECOLOR mistakes the static foreground into the background model. In contrast, COROLA shows acceptable results for these challenging sequences because using GMM and the difference between outliers and the rest of pixels is boosted and so COROLA can detect intermittently moving objects better than DECOLOR.

To show the capability of COROLA, we include “OR-PCA+MRF” [28] in our evaluation; however, since this approach does not use an optimization framework, it does not perform as well as COROLA. Table 2 shows that our method outperforms the state-of-the-art.

4.2.2. Computational Cost

COROLA is implemented in Matlab and C++. We run all experiments on a PC with a 3.4 GHz Intel i7 CPU and 16 GB RAM. To show the role of online methods on continuous operation we compare the scalability of COROLA with DECOLOR under varying spatial resolution and the number of images.

Unlike DECOLOR, the computational cost of COROLA is independent of
Figure 6: The results of COROLA on 6 sequences from three datasets Change Detection, I2R, and Wallflower. Columns (a) and (b) show the original query image and the ground truth (GT) for the foreground. Columns (c) and (f) show the results of COROLA for estimating the background $L$, and the detected foreground objects $S$, respectively. Columns (d) and (e) show intermediate results for outliers $E$, and $\hat{E}$, respectively.

the number of images because the dominant cost of DECOLOR comes from the computation of SVD in each iteration, and by increasing the size of matrix $D$, the computation time grows at least linearly with respect to the number of images. We compare the computation time of COROLA with DECOLOR after convergence of both methods in Table 3. In this table, the average time for processing of each frame by DECOLOR method increases where it is an order of magnitude slower than COROLA with only 1000 images.
Table 2: Comparison of F-measure score

| Sequence   | MOG  | SSGoDec | DECOLOR | ORPCA+MRF | COROLA |
|------------|------|---------|---------|-----------|--------|
| Water surface | 0.7723 | 0.4473 | 0.9022 | 0.9166 | 0.9503 |
| Fountain   | 0.7766 | 0.2574 | 0.2075 | 0.8283 | 0.9175 |
| Curtain    | 0.7709 | 0.4344 | 0.8700 | 0.8920 | 0.9038 |
| Canoe      | 0.5114 | 0.3091 | 0.1603 | 0.8534 | 0.8901 |
| Overpass   | 0.5095 | 0.5517 | 0.3573 | 0.8272 | 0.8471 |
| Waving trees | 0.6639 | 0.1829 | 0.8845 | 0.8689 | 0.8688 |
| Hall       | 0.6802 | 0.5713 | 0.8169 | 0.7844 | 0.8298 |

Table 3: Time evaluation of COROLA with DECOLOR method

| Methods   | Resolution × #images | Low Rank (s) | MRF (s) | Total (s) |
|-----------|----------------------|--------------|---------|-----------|
| DECOLOR   | [320 × 240] × 200    | 0.1036       | 0.0828  | 0.1864    |
|           | [320 × 240] × 400    | 0.1531       | 0.1297  | 0.2828    |
|           | [320 × 240] × 600    | 0.1687       | 0.1601  | 0.3279    |
|           | [320 × 240] × 800    | 0.2016       | 0.1825  | 0.3841    |
|           | [320 × 240] × 1000   | 0.3948       | 0.3191  | 0.7139    |
| COROLA    | [320 × 240] × 1000   | 0.0231       | 0.0605  | 0.0836    |

Scalability in spatial resolution is another advantage of online method against batch processing methods, where increasing the resolution of images significantly affects DECOLOR method. Using high dimensional images results in a huge matrix $D$ and so decomposing $D$ becomes very expensive. COROLA is an online method and is independent from the number of images, therefore we do not have a large $D$ and its computation time grows linearly with the image resolution.

4.3. Experiments on a Moving Camera

In this section, we test our method on real public sequences for moving cameras namely “Berkeley motion segmentation dataset” [38]. Table 4 shows
the details of five challenging sequences that we use in our experiments.

Table 4: Details of all sequences used in our experiments for moving camera

| Dataset                              | Sequences   | Size × #frames |
|--------------------------------------|-------------|----------------|
| cars6                                | [320,240] × 30 |
| cars7                                | [320,240] × 24 |
| Berkeley motion segmentation         | people1     | [320,240] × 40 |
|                                      | tennis      | [320,240] × 200 |
|                                      | marple13    | [320,240] × 75 |

Fig. 7 shows the qualitative results of COROLA in comparison with DECOLOR method for moving object detection using a moving camera. First

Figure 7: Comparison of foreground objects between DECOLOR and COROLA. columns (a) and (b) show the input image and its ground truth. columns (c) and (d) show the obtained foreground mask for DECOLOR and COROLA methods, respectively.
three experiments have been performed on short sequences “cars6”, “cars7”, “people1” and the results from COROLA are comparable with those from DECOLOR method. For the last two sequences “marple13” and “Tennis”, DECOLOR has a problem to align images when the last images are not similar with the first images of these sequences. This is common in continuous processing and all of batch methods have problem with this. To show the result of DECOLOR on marple13 and tennis sequences (in the last two rows of Fig. 7), we used last 30 images of the sequences, which have less camera motion. As we mentioned before, in continuous processing with a moving camera, the last images in the sequence are no longer similar to the initial ones in matrix $D$ and therefore, any attempt to find a low-rank matrix approximation is not appreciate and doomed to fail. In contrast, since COROLA works online and only considers the last two images it can process the last two sequences of Table 4 without any problems and provides acceptable results in comparison with the DECOLOR method and an online registration based method in [9].

Table 5 shows the quantitative evaluation of COROLA in comparison with DECOLOR and the method presented in [9]. Experiments over all five sequences show that the results of COROLA method is comparable with the DECOLOR method and also has the advantage of online methods for real-time continuous processing. Note that with more than 30 images in the sequence, DECOLOR can no longer produce a valid result due to the significant dissimilarity of the images later in the sequence from the initial ones. In contrast, our sequential method is always able to produce a valid result often with higher accuracy.

5. Conclusion

In this paper, we have proposed a novel online method named COROLA to detect moving objects in a video using the framework of low-rank matrix approximation. Our online framework works iteratively on each image of the video to extract foreground objects accurately. The key to our online formulation is to exploit the sequential nature of a continuous video of a scene where the back-
ground model does not change discontinuously and can therefore be obtained by updating the background model learned from preceding images. We have applied COROLA to the case of a moving camera. Since our method works online and is independent of the number of images, it is suitable for real-time object detection in continuous monitoring tasks. Our method overcomes the problems of batch methods in terms of memory storage, time complexity, and camera motion. Also important to the success of COROLA is using Gaussian model to separate noise from outliers and also to tune the costs of assigning labels in MRF via \( \sigma \) and weights of Gaussian parameters, dynamically and automatically especially when the object moves very slow or stops for some frames. According to our extensive experiments on synthetic data and real data sequences, COROLA archives the best performance in comparison with all evaluated methods in our experiments.

Despite its satisfactory performance in all of our experiments, COROLA shares one disadvantage with DECOLOR. Since both methods have non-convex formulations, they might be converged to a local minimum with results depending on initialization of parameters; however, for the case of background modeling, images are roughly similar and parameters do not change significantly. Therefore, the issue of local minimum has not affected successful object detection in our experiments.

**Acknowledgements** This work was supported by the Natural Sciences

| Sequence   | FFD based model | DECOLOR | COROLA |
|------------|-----------------|---------|--------|
| cars6      | 0.8870          | 0.9052  | 0.9409 |
| cars7      | 0.8257          | 0.8441  | 0.8867 |
| people1    | 0.8122          | 0.9666  | 0.9056 |
| tennis     | 0.8494          | 0.8404 \(^1\)/NA | 0.8642 |
| marple13   | 0.6407          | 0.8063 \(^1\)/NA | 0.8271 |

Table 5: Comparison of F-measure score. \(^1\) last 30 images is used.
and Engineering Research Counsel (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN) and by Alberta Innovates Technology Future (AITF).

References

[1] Y. Tian, A. Senior, M. Lu, Robust and efficient foreground analysis in complex surveillance videos, in: Journal of Machine Vision and Applications, volume 23, 2012, pp. 967–983.

[2] M. Shakeri, H. Deldari, H. Foroughi, A. Saberi, A. Naseri, A novel fuzzy background subtraction method based on cellular automata for urban traffic applications, in: International Conference on Signal Processing (ICSP), 2008, pp. 899–902.

[3] L. Unzueta, M. Nieto, A. Cortes, J. Barandiaran, O. Otaegui, P. Sanchez, Adaptive multi-cue background subtraction for robust vehicle counting and classification., Intelligent Transportation Systems, IEEE Transactions on 13 (2012) 527–540.

[4] M. Shakeri, H. Zhang, Real-time bird detection based on background subtraction, in: World Congress on Intelligent Control and Automation (WCICA), 2012, pp. 4507–4510.

[5] C. Stauffer, W. Grimson, Adaptive background mixture models for real-time tracking, in: Computer Vision and Pattern Recognition, 1999 IEEE Computer Society Conference on, volume 2, 1999, pp. 246–252.

[6] A. Elgammal, R. Duraiswami, D. Harwood, L. S. Davis, Background and foreground modeling using nonparametric kernel density estimation for visual surveillance, Proceedings of the IEEE 90 (2002) 1151–1163.

[7] Y. Sheikh, O. Javed, T. Kanade, Background subtraction for freely moving cameras, in: Computer Vision(ICCV), 2009 IEEE 12th International Conference on, 2009, pp. 1219–1225.
[8] A. Elqursî, A. Elgammal, Online moving camera background subtraction, in: Computer Vision, ECCV2012, 2012, pp. 228–241.

[9] M. Shakeri, H. Zhang, Detection of small moving objects using a moving camera, in: Intelligent Robots and Systems(IROS 2014), 2014 IEEE/RSJ International Conference on, 2014, pp. 2777–2782.

[10] D. Cremers, S. Soatto, Motion competition: a variational approach to piecewise parametric motion segmentation, International Journal of Computer Vision 62 (2005) 249–265.

[11] R. Vidal, Y. Ma, A unified algebraic approach to 2d and 3d motion segmentation, in: In European Conference on Computer Vision (ECCV), 2004, pp. 1–15.

[12] P. Ochs, T. Brox, Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions, in: Computer Vision (ICCV), IEEE International Conference on, 2011.

[13] R. Vidal, Subspace clustering, IEEE Signal Processing Magazine 28 (2011) 52–68.

[14] M. Piccardi, Background subtraction techniques: A review, in: Systems, Man and Cybernetics, 2004 IEEE International Conference on, volume 4, 2004, pp. 3099–3104.

[15] C. Wren, A. Azarbeyjani, T. Darrell, A. Pentland, Pfinder: real-time tracking of the human body, Pattern Analysis and Machine Intelligence, IEEE Transaction on 19 (1997) 780–785.

[16] C. Stauffer, W. Grimson, Adaptive background mixture models for real-time tracking, in: Computer Vision and Pattern Recognition (CVPR), IEEE conference on, 1999.

[17] A. Elgammal, D. Harwood, L. Davis, Non-parametric model for background subtraction, in: European Conference on Computer Vision (ECV), 2000.
[18] X. Cui, J. Huang, S. Zhang, D. Metaxas, Background subtraction using low rank and group sparsity constraints, in: In European Conference on Computer Vision (ECCV), 2012, pp. 612–625.

[19] E. Candes, X. Li, Y. Ma, J. Wright, Robust principal component analysis?, Journal of the ACM (JACM) 58 (2011).

[20] Z. Tianyi, D. Tao, Greedy bilateral sketch, completion and smoothing, in: Artificial Intelligence and Statistics, International Conference on, 2013, pp. 650–658.

[21] T. Bouwmans, E. Zahzah, Robust pca via principle component pursuit: A review for a comparative evaluation in video surveillance, Computer Vision and Image Understanding 122 (2014) 22–34.

[22] Y. Shen, Z. Wen, Y. Zhang, Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization, in: TR11-02, Rice University, Houston, TX, 2011.

[23] Z. Lin, R. Liu, Z. Su, Linearized alternating direction method with adaptive penalty for low rank representation, in: Advances in Neural Information Processing Systems (NIPS), 2011.

[24] J. Cai, E. J. Candes, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. on Optimization 20 (2010) 1956–1982.

[25] R. Cabral, F. L. Torre, J. Costeira, A. Bernardino, Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition, in: Computer Vision (ICCV), IEEE International Conference on, 2013, pp. 2488–2495.

[26] Z. Tianyi, D. Tao, Godec: Randomized low-rank and sparse matrix decomposition in noisy case, in: Machine Learning, 2011 International Conference on, 2011, pp. 33–40.
[27] X. Zhou, C. Yang, W. Yu, Moving object detection by detecting contiguous outliers in the low-rank representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 597–610.

[28] J. Feng, H. Xu, S. Yan, Online robust pca via stochastic optimization, in: Advances in Neural Information Processing Systems (NIPS), 2013, pp. 404–412.

[29] S. Javed, S. Oh, A. Sobral, T. Bouwmans, S. Jung, Or-pca with mrf for robust foreground detection in highly dynamic backgrounds, in: Asian Conference on Computer Vision (ACCV), 2014.

[30] R. Mazumder, T. Hastie, R. Tibshirani, Spectral regularization algorithms for learning large incomplete matrices, Journal of Machine Learning Research 11 (2010) 2287–2322.

[31] S. Li, Markov random field modeling in image analysis, Springer Verlag (2009).

[32] Z. Zivkovic, Improved adaptive gaussian mixture model for background subtraction, in: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 2, 2004, pp. 28–31.

[33] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, Pattern analysis and machine intelligence, IEEE Transaction on 23 (2001) 1222–1239.

[34] V. Kolmogorov, R. Zabih, What energy function can be minimized via graph cuts?, Pattern analysis and machine intelligence, IEEE Transaction on 26 (2004) 147–159.

[35] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, P. Ishwar, Changeddetection.net: a new change detection benchmark dataset, in: Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference on, 2012, pp. 1–8.
[36] L. Li, W. Huang, I. Gu, Q. Tian, Statistical modeling of complex backgrounds for foreground object detection, Image Processing, IEEE Transaction on 13 (2004) 1459–1472.

[37] K. Toyama, J. Krumm, B. Brumitt, B. Meyers, Wallflower: principles and practice of background maintenance, in: International Conference on Computer Vision (ICCV), 1999, pp. 255–261.

[38] R. Tron, R. Vidal, A benchmark for the comparison of 3d motion segmentation algorithms, in: Computer Vision,IEEE International Conference on, 2007.