ADAPTIVE BACKGROUND MODELING WITH TEMPORAL FEATURE UPDATE FOR DYNAMIC FOREGROUND OBJECT REMOVAL

by

Li Yin

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Jacob Gunther Major Professor	Charles M. Swenson Committee Member
Don Cripps Committee Member	Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2016

ABSTRACT

Adaptive Background Modeling with Temporal Feature Update for Dynamic Foreground Object Removal

by

Li Yin, Master of Science

Utah State University, 2016

Major Professor: Jacob Gunther
Department: Electrical and Computer Engineering

In the study of computer vision, background modeling is a fundamental and critical task in many conventional applications. This thesis presents an introduction to background modeling and various computer vision techniques for estimating the background model to achieve the goal of removing dynamic objects in a video sequence.

The process of estimating the background model with temporal changes in the absence of foreground moving objects is called adaptive background modeling. In this thesis, three adaptive background modeling approaches were presented for the purpose of developing "teacher removal" algorithms. First, an adaptive background modeling algorithm based on linear adaptive prediction is presented. Second, an adaptive background modeling algorithm based on statistical dispersion is presented. Third, a novel adaptive background modeling algorithm based on low rank and sparsity constraints is presented. The design and implementation of these algorithms are discussed in detail, and the experimental results produced by each algorithm are presented. Lastly, the results of this research are generalized and potential future research is discussed.

(78 pages)

PUBLIC ABSTRACT

Adaptive Background Modeling with Temporal Feature Update for Dynamic Foreground Object Removal

Li Yin

This thesis explores several approaches in order to develop an effective algorithm for separating the moving objects from the background so that the background alone may be displayed in a video sequence. In particular, this thesis seeks to develop an algorithm that can successfully remove a moving teacher from the whiteboard, so that the writing on the whiteboard may be fully visible to the audience.

In this thesis, some prior works related to this problem are studied to understand the basic principles of adaptive background modeling, or removing the foreground from a background with occasional changes in a video sequence. Three different algorithms are developed for this purpose. The design and implementation of these algorithms are discussed in detail, and the results are presented.

Lastly, the results of this research are generalized and potential future research is discussed.

To my family...

ACKNOWLEDGMENTS

First, I would like to thank my major professor Dr. Jacob Gunther as well as my committee for providing invaluable support, guidance, and insight.

I would also like to thank all of the professors in the Electrical and Computer Department at Utah State University for teaching me everything I know.

I would like to thank my family for their unwavering support and encouragement, as well as my friends and colleagues Trevor Landeen, David Neal, Andrew Pound, and Mehedi Hasan for their assistance, advice, and friendship over the course of my attendance at Utah State University.

Last, but not least, I would like to thank Utah State University's Distance Education Program for providing the opportunity to conduct this research.

Li Yin

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

ACRONYMS

IIR	infinite impulse response
FIR	finite impulse response
MOG	mixture of Gaussians (same as GMM)
GMM	Gaussian mixture model
EM	expectation maximization
DSP	digital signal processing
MSE	mean square error
LMS	least mean squares
AR	auto-regressive
MA	moving-average
PPM	portable pix-map
EWMA	exponentially weighted moving-average
RGB	red-green-blue (color space or color model)
FPS	frame per second (frame rate)
SVD	singular value decomposition
PCA	principal component analysis
RPCA	robust principal component analysis
RPCA-PG	robust principal component analysis via proximal gradient
RPCA-PCP	robust principal component analysis via principal component pursuit
FPGA	field-programmable gate array
GPU	graphics processing unit

CHAPTER 1

Introduction

The removal of moving objects from complex background scenes in a video sequence is a challenging task because of the uncertainties of foreground and background dynamics. For instance, it is common that in a given video sequence there are moving objects in the foreground that become stationary and are added to the background. Conversely, there are stationary objects in the background that start to move, and are added to the foreground.

This research studied various image processing and computer vision techniques to estimate the background with occasional changes in the absence of moving objects in the foreground. In this research, three approaches that accomplished adaptive background modeling were studied and implemented to separate dynamic objects in the foreground from the adaptive background.

1.1 Adaptive Background Modeling

In the field of computer vision, moving objects within a video sequence are classified as the foreground model, static objects within a video sequence are classified as the background model, and the background model with minute or temporal changes is classified as the adaptive background model. The separation of the foreground model from the background is a critical task in many computer vision applications, such as human detection and tracking, surveillance, traffic monitoring, and so on. A common approach to achieve such separation is to perform background subtraction, where each frame in a video sequence is compared with or subtracted from a background model. The resulting differences in pixels are refer to as moving objects, or the foreground. Such comparison is highly dependent on an estimation of the background model, and this makes background modeling particularly important. However, conventional background modeling methods are undoubtedly affected by minute variations in the background model, for instances, the starting and the stopping of a moving

object, weather, and illumination changes. The study of adaptive background modeling can be significant for solving these problems. Furthermore, adaptive background modeling can be useful for addressing the problem discussed in the next section.

1.2 Thesis Application

Consider a scenario in a classroom where the teacher is writing on the whiteboard and students are taking notes as the teacher writes. The teacher often needs to stand in front of his writing while writing on a different part of the whiteboard. Meanwhile, students are standing up, moving their heads left and right, and trying to find appropriate perspectives to see notes that are covered by the teacher. However, it is not easy for students who are taking online classes to get a new perspective. Would it be possible to remove the teacher, but keep the notes? Another way of asking this question is, would it be possible to see the notes without seeing the teacher? In this respect, the "teacher removal" algorithm behaves in a similar manner as a smart whiteboard. However, the algorithms have potential advantages in terms of expenses.

The research is dedicated to develop a "teacher removal" algorithm for the distance education program at Utah State University. As a result of this research, the algorithm will allow distance learners to read notes on whiteboards without physical interference caused by teachers' movements.

1.3 Thesis Outline

In Chapter 2, some prior works and the studies of existing background modeling techniques are reviewed. In Chapter 3, the proposed approaches are described in details to explain the adaptive background modeling algorithms. In particular, three different models are presented for the purpose of adaptive background modeling. In Chapter 4, the experimental data and the performance of the different adaptive background modeling algorithms are presented and discussed in details. Finally, the development of this research is concluded, and future work is discussed in Chapter 5.

CHAPTER 2

Prior Work

While numerous non-adaptive background modeling techniques have been investigated in the past, most researchers have moved on to adaptive techniques because of their flexibility. Specifically, adaptive background modeling techniques are able to handle different classes of variations in practice. In this research, four different existing techniques were studied to understand some of the basic principles of adaptive background modeling. In the first section, different classes of adaptive background modeling techniques were categorized and discussed. In the second section, some of the most common existing techniques, as well as a recently developed techniques, were reviewed.

2.1 Classification of Existing Techniques

Adaptive background modeling techniques could be classified into two general categories – non-recursive techniques and recursive techniques, according to Cheung and Kamath [1]. A sliding window type of approach is applied to the non-recursive techniques to achieve the goal of adaptive background modeling. In particular, the non-recursive techniques use past frames that are stored in the buffers to estimate the background of these frames on a pixel-level. The advantage of these techniques is that the future estimations do not rely on past estimations after the buffers are updated, thus these techniques can be robust to fast temporal changes. However, the uses of buffers can be the drawback of these techniques if the video sequence has a high frame rate. Conversely, sliding windows or buffers are not required for recursive background modeling techniques. Instead, the estimations are updated recursively for each new frame based on feedback. The errors from previous estimations can accumulate over time or propagate throughout time, resulting in adaptations being slow and estimations being not robust to fast temporal changes.

Other techniques take an alternative approach to achieve adaptive background modeling by applying matrix decomposition techniques. For instance, some novel adaptive background modeling techniques transform a video sequence into a data matrix, and the background and the foreground is decomposed by applying principal component analysis (PCA) types of transformations. Technically, this type of technique can be classified as a type of recursive technique, because of the use of iterative methods. The advantage of this type of technique is the accuracy of the separation. However, one of the disadvantages is the necessity of some complex mathematical operations.

2.2 Literature Survey

In this section, four different existing techniques were reviewed to prepare for the development of the adaptive background modeling algorithms. In particular, one non-recursive techniques, two recursive techniques, and a novel techniques were studied in this research.

2.2.1 Non-recursive Techniques

Linear Predictive Filter. A linear predictive filter based technique was proposed by Toyama et al. [2], and this technique applies an infinite impulse response (IIR) Wiener filter on a pixel-level to estimate adaptive background model. The estimated pixel-level background models are then segmented as a region-level background model by grouping neighboring pixels based on the histograms of the pixels. The frames in a video sequence are stored in a buffer for the estimations, and IIR filter coefficients are calculated for each new frame. The needs of recalculating the histograms and the IIR filter coefficients for each new frame results in this technique being hard to implement in real time.

2.2.2 Recursive Techniques

Kalman Filter. A Kalman filter is a recursive linear quadratic estimator which has a wide range of applications in different areas of study in engineering. In this research, a Kalman filter was investigated to achieve the goal of adaptive background modeling.

A Kalman filter was applied on a pixel-level to estimate the adaptive background model without prior measurement of the ground-truth. Many different versions of Kalman filter based adaptive background modeling algorithm have been proposed in the past. One of the simplest version was proposed by Scott et al. [3]. In this approach, the Kalman filter modeled each pixel as single Gaussian, and by alternating the mean and standard deviation update equations, the Kalman filter was able to predict and update the background model for each pixel. The pixel-level background models are then segmented by applying morphological close operations.

Mixture of Gaussians. Mixture of Gaussians (MOG) or Gaussian Mixture Model (GMM) is one of the most common adaptive background modeling algorithms. A GMM is a weighted probability density function with the parameters of weight, mean, and covariance matrix. These parameters are often initialized and maintained from given training data using a iterative expectation maximization (EM) algorithm [4]. However, after performing preliminary tests with GMM, it was evident that using the iterative EM algorithm to estimate GMM parameters could be time consuming, and difficult to implement in real-time. Instead, parameters are updated and adapted using an online K-mean algorithm for real-time consideration [5]. Moreover, in the preliminary tests with GMM, it failed to adapt and capture fast temporal changes in the background. However, recent research suggests that improved GMM is capable of capturing temporal changes in the background [6, 7].

2.2.3 Other Techniques

Low Rank and Sparsity Constraints. Low rank and sparsity constraints is a novel background subtraction method which applies matrix decomposition techniques to extract the anomalies in a video sequence. A matrix was formed by stacking up all the tracked points in a video sequence. This matrix was then decomposed into a background matrix and a foreground matrix based on two sparsity constraints, with the background matrix being low rank, and the foreground matrix being sparse. An implementation of low rank and sparse based model was proposed by Cui et al. [8]. However, this method relies on an existing dense point tracker to generate the trajectories of the moving objects [9,10]. In this

research, similar techniques were studied without the implementation of the existing point tracker. Instead, an innovative method based on similar matrix decomposition techniques was studied and implemented [11, 12]. By applying the same fundamental idea, a matrix was formed by stacking all the vectorized frames in a video sequence. This data matrix was then decomposed into a background low rank matrix and a foreground sparse matrix by an iterative method. The vectorized frames in the resulting low rank matrix and sparse matrix were then transformed back into a video sequence.

CHAPTER 3

Proposed Approaches

In this chapter, three adaptive background modeling approaches are presented in different sections. In the first section, a non-recursive approach based on adaptive linear prediction is presented. In the second section, a simple approach based on statistical dispersion is presented. In the third section, an approach based on low rank and sparsity constraints and matrix decomposition is presented. While the first two approaches share similarities with some existing methods, the third approach is a novel technique to achieve the goal of adaptive background modeling. The ideas behind these approaches are further discussed in each section, as well as the algorithms and the implementation details.

Some helpful notations in this chapter:

- Row coordinate: i
- Column coordinate: j
- Color channel coordinate: k
- Discrete time or frame time: n
- Number of rows: I
- Number of columns: J
- Number of color channels: K
- Number of frames: Q
- Frames in a video sequence: $F(\cdot, \cdot, \cdot, \cdot)$

3.1 Adaptive Linear Prediction Based Model

Adaptive filters are widely used in many digital signal processing (DSP) applications due to their ability of "learning" the relationship between signals. One common adaptive filter application is adaptive linear prediction, also known as forward prediction. The purpose of adaptive linear predictor is to use an adaptive filter to estimate or predict the future values of a signal based on past values of the signal. In this section, the fundamentals of adaptive filtering and the least mean square (LMS) adaptive algorithm [13] are explained to introduce the usage of the adaptive linear prediction for background modeling.

3.1.1 LMS Adaptive Filter and Prediction Error

One class of adaptive filters is based on the theory of optimal filtering. The objective of the optimal filter is to minimize the mean square error (MSE) between two input signals, a reference signal $x(n)$ and a desired signal $d(n)$. The filter output $y(n)$ is a linear combination of the past input samples. An error signal $e(n)$ is produced by taking the difference of the desired signal $d(n)$ and the output signal $y(n)$. The adaptive algorithm adjusts the filter coefficients iteratively to minimize the error in a mean square sense. When the power of error signal $e(n)$ converges to a minimal value, the input signal $x(n)$ becomes orthogonal to the error signal $e(n)$ in the signal space. The convergence indicates that $x(n)$ can be uniquely expressed as a linear combination of $y(n)$ plus the uncorrelated term $e(n)$. The general structure of an adaptive filter is shown in Figure. 3.1.

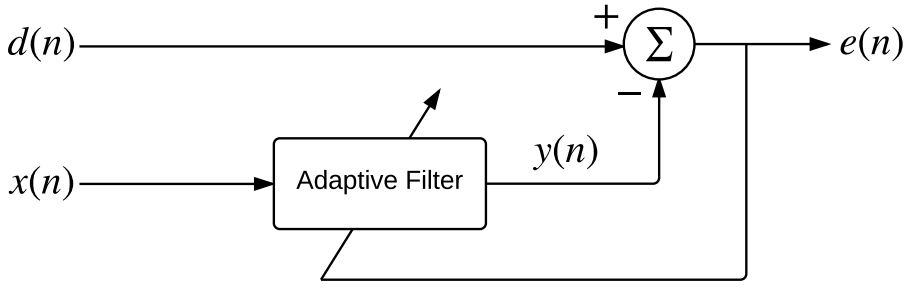


Fig. 3.1: The general structure of an adaptive filter.

The general structure of an adaptive filter is configured slightly different to achieve adaptive linear prediction. The desired signal is taken as the system input $d(n) = x(n)$, and the adaptive filter input is a delayed version of the system input $x(n - \Delta)$. In this configuration, the adaptive filter predicts the current sample of the system input $x(n)$ as a linear combination of the past q input samples $[x(n-\Delta), x(n-\Delta-1), \ldots, x(n-\Delta-q+1)]$, where q is the filter length. Similar to the general structure of an adaptive filter, the outputs of an adaptive linear predictor are the error signal $e(n)$ and the filter output $y(n)$. The structure of an adaptive linear predictor is shown in Figure. 3.2.

The adaptive algorithm is at the heart of any adaptive filter application, and one of the most common adaptive algorithm is the LMS algorithm. The LMS algorithm is an approximation of the steepest descent method where the expectation operator of the MSE $\mathbb{E}[e^2(n)]$ is ignored. The LMS algorithm is derived as follows. The error signal $e(n)$ at time n is given by:

$$e(n) = d(n) - y(n) \tag{3.1}$$

The output signal at time n is given by:

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n) = \sum_{i=0}^{q-1} w_i(n)x(n-i) \tag{3.2}$$

where q is the filter length, $\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-q+2), x(n-q+1)]^T$, and $\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{q-2}(n), w_{q-1}(n)]^T$.

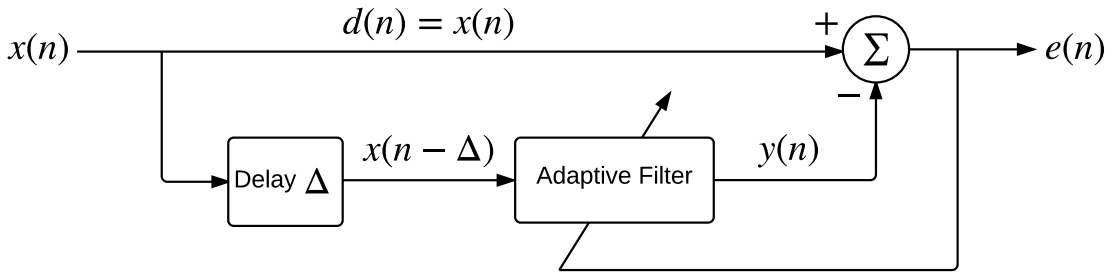


Fig. 3.2: The structure of an adaptive linear predictor.

Now, taking the partial derivative of error signal $e^2(n)$ with respect to the filter coefficients $\mathbf{w}(n)$.

$$e^2(n) = (d(n) - \mathbf{w}^T(n)\mathbf{x}(n))^2 = (d(n) - \sum_{i=0}^{q-1} w_i(n)x(n-i))^2$$

$$\frac{\partial e^2(n)}{\partial \mathbf{w}(n)} = -2e(n)\mathbf{x}(n)$$

The filter update at time $n+1$ is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \frac{\partial e^2(n)}{\partial \mathbf{w}(n)}$$

Hence, the filter update at time $n+1$ becomes $\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$, or

$$w_i(n+1) = w_i(n) + 2\mu e(n)x(n-i) \tag{3.3}$$

where μ is the step size, it is a constant that determines the rate of convergence.

3.1.2 Adaptive Background Modeling Algorithm

The adaptive linear prediction works well under the influence of the noise introduced by the camera because the pixel intensity varies slightly. But the adaptive linear prediction produces large a prediction error when random changes occur to a pixel. Based on this observation, the prediction error could be used as a threshold to separate the changing foreground pixels from the static background pixels.

In this approach, a video sequence is considered as an all-zero or a moving-average (MA) model. The LMS finite impulse response (FIR) adaptive linear filters are performed on a pixel-level throughout frames to capture significant prediction errors that are caused by random pixel intensity changes. Furthermore, the predicted pixel intensity of a given pixel is relatively close to the actual pixel intensity, and the prediction error is relatively insignificant when there are no moving objects present at this pixel. However, when moving objects pass through a given pixel, the predicted pixel intensity is different from the actual

pixel intensity, and the prediction error is relatively large.

For example, Figure. 3.3, Figure. 3.4, and Figure. 3.5 are the adaptive linear prediction simulation results of a particular pixel on red, green, and blue color channels. In this simulation, the LMS FIR adaptive linear prediction used $q = 30$ filter coefficients, $\Delta = 10$ frames of delay, and a step size of $\mu = 0.06$. The simulation pixel was taken from the video *visiontraffic.avi* in MATLAB [14] vision demos at pixel location $F(i, j) = F(150, 550)$ for every color channel throughout all frames. In Figure. 3.3, Figure. 3.4, and Figure. 3.5, large prediction errors at the beginning of the simulation were caused by the initializations of the adaptive filters. Upon the convergence of the adaptive filters, the predicted pixel intensities were very close to the actual pixel intensities, thus the prediction errors were insignificant. However, the pixel intensities decreased sharply at frame time $n = 320$. This was caused by a moving object entering this pixel location. Furthermore, the dramatic changes in pixel intensities resulted in significant differences between actual pixel intensities and predicted pixel intensities, thus resulting significant prediction errors.

A system based on adaptive linear prediction is equipped with buffers and finite state machines to achieve the goal of adaptive background modeling. First, the system stores previous pixel values at past frame time $n - (q + \Delta)$ in buffers for each pixel location. Second, the system uses delayed pixel values to perform probabilistic predictions, and LMS FIR filter coefficients are updated for every new frame. The system then toggles between two states, "hold pixel" or "update pixel" based on $\sqrt{e^2(n)}$ at current frame time n. If the prediction error at a particular pixel is greater than a threshold T_{hold}, then the state of this pixel becomes "hold pixel", and the system will take the previous pixel value in the buffer as an output. Furthermore, if the prediction error of a particular pixel is less than a threshold T_{update}, the state of this pixel becomes "update pixel", and the system will take the current pixel value as an output. The thresholds T_{hold} and T_{update} are hand-picked values based on a wide range of tests. $T_{hold} = 15$ and $T_{update} = 2$ worked for most tested video sequences in this research. The overall design of the adaptive linear prediction background modeling algorithm is given in Algorithm. 3.1.

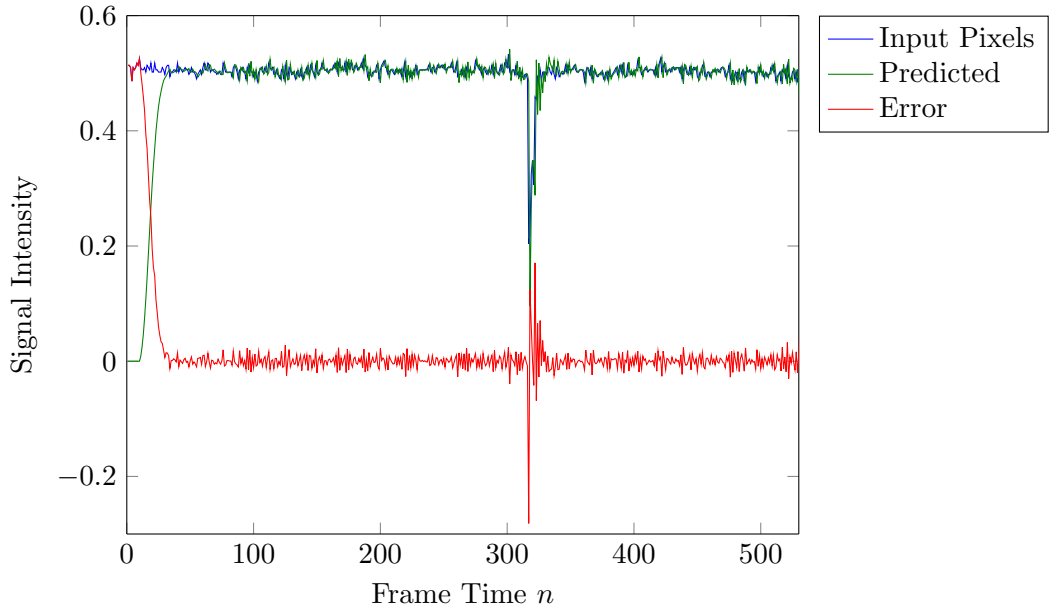


Fig. 3.3: The adaptive linear prediction result of a pixel in red color channel.

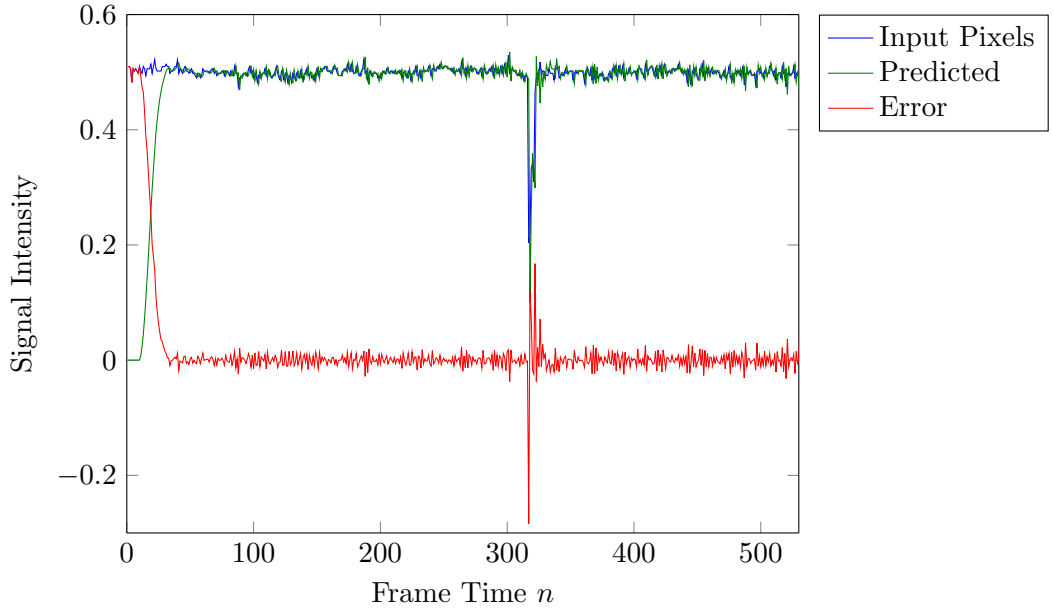


Fig. 3.4: The adaptive linear prediction result of a pixel in green color channel.

Algorithm 3.1 Adaptive Linear Prediction Background Modeling Algorithm

Input:

Video sequence $F(I, J, K, Q)$,
Thresholds T_{update} and T_{hold},
Filter length q,
Frame delay Δ,
Step size μ

Output:

Background model video sequence $F_{bg}(I, J, K, Q)$

begin

Initialize LMS FIR filter coefficients $W(I, J, K, q)$
Initialize states $S(I, J, K, Q)$
for $\{i = 1 : I\}$
  for $\{j = 1 : J\}$
    for $\{k = 1 : K\}$
      for $\{n = 1 : Q\}$
        Execute LMS FIR filters with $F(i, j, k, n)$
        Calculate prediction errors $e(i, j, k, n)$
        Update LMS FIR filter coefficients $W(i, j, k, q)$
        Calculate error thresholds $T = \sqrt{e^2(i, j, k, n)}$
        Update buffer $F_{past}(i, j, k, n)$ from $F(i, j, k, n - (q + \Delta))$
        if $\{T >= T_{hold}$ **and** $S(i, j, k, n - 1) = 0\}$
          $S(i, j, k, n) = 1$
          Output background model $F_{bg}(i, j, k, n)$ from $F_{past}(i, j, k, n)$
        end
        if $\{T <= T_{update}$ **and** $S(i, j, k, n) = 1\}$
          $S(i, j, k, n) = 0$
          Output background model $F_{bg}(i, j, k, n)$ from $F(i, j, k, n)$
        end
        Update states $S(i, j, k, n - 1) = S(i, j, k, n)$
      end
    end
  end
end

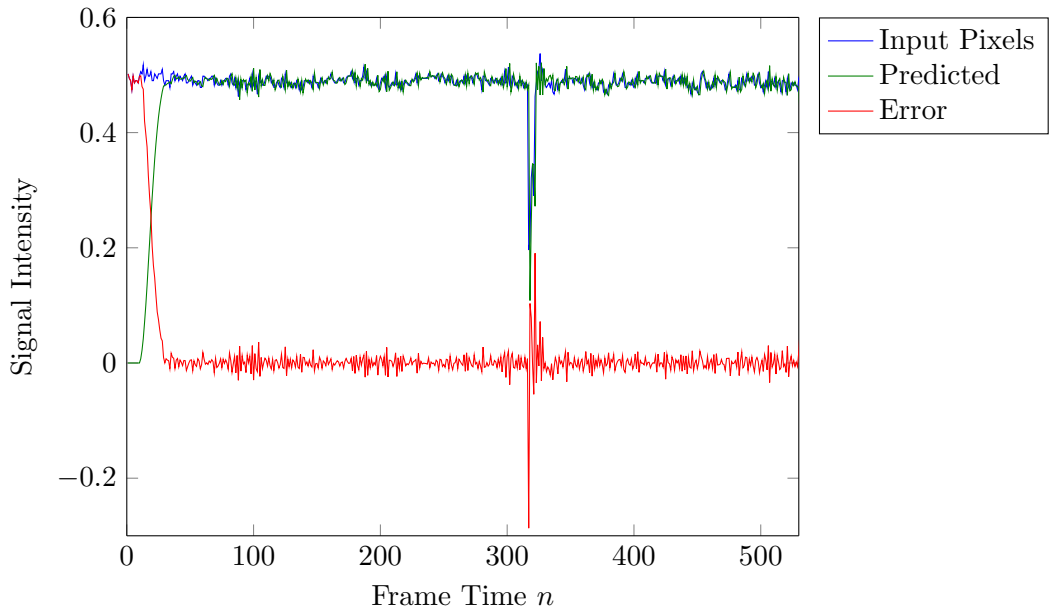


Fig. 3.5: The adaptive linear prediction result of a pixel in blue color channel.

3.1.3 Notes on Implementation

In this research, most tested video sequence only required $q = 10$ filter coefficients, $\Delta = 20$ frames of delay, and a step size of $\mu = 1e^{-6}$ to achieve the goal of adaptive background modeling. Because of this advantage, the adaptive linear prediction background modeling algorithm can be realized in real-time for future implementations. However, a similar approach proposed by Toyama et al. [2] used infinite impulse response (IIR) Wiener filter as the predictive technique. This technique requires more intensive computation to solve $p = 30$ filter coefficients based on sample covariance for auto-regressive (AR) models at each frame time. Such complex computation makes this technique difficult to realize in real-time.

Before processing, the input video sequence was converted to portable pix-map (PPM) format in the implementation of this approach. After this conversion, a PPM file stored all the frames in a video sequence as a section of contiguous memory in the physical memory. The purpose of using PPM format is to ensure all the frames are available for the process, and to avoid the overheads that are caused by reading in video files during run-time. However, some long duration video sequences may occupy a tremendous amount of the physical

memory, and this can cause problems while allocating memory for such video sequences. Therefore, PPM format may not be suggested if a video sequence has a substantial amount of frames.

3.2 Statistical Dispersion Based Model

In statistics, dispersion is a measurement of variability for any given statistical sample, and it describes the tendency of a distribution to be scattered or stretched based on the measures of dispersion. Some common measures of statistical dispersion are the range, variance, and standard deviation. In this section, a simple implementation and a modified implementation of statistical dispersion background modeling algorithm are presented.

3.2.1 EWMA Filter and Dispersion

In this approach, the input video sequence is filtered by exponentially weighted moving-average (EWMA) IIR filter on a pixel-level, and the variance of filtered output is used as the measure of statistical dispersion to distinguish the foreground changing pixels from the background static pixels. The EWMA filter is used as a pixel-level low-pass filter to smooth out the high frequency noise based on a weighted previous output and a weighted current input. First, the EWMA filter smooths out the high frequency noise in the input signal $x(n)$, and the output $y_1(n)$ at time n is given by:

$$y_1(n) = (1 - \lambda)y_1(n - 1) + \lambda x(n) \tag{3.4}$$

where λ is the smoothing factor, and $0 < \lambda < 1$.

Second, the EWMA filter smooths out the high frequency noise in the squared input signal $x^2(n)$, and the output $y_2(n)$ at time n is given by:

$$y_2(n) = (1 - \lambda)y_2(n - 1) + \lambda x^2(n) \tag{3.5}$$

where λ is the smoothing factor, and $0 < \lambda < 1$.

Third, the estimation of the statistical dispersion is based on the variance of input $x(n)$, which can be expressed as $\mathbb{E}[x^2(n)] - \mathbb{E}[x(n)]^2$, or the difference between two EWMA filtered outputs ${y_1}^2(n)$ and $y_2(n)$. The comparison would only make sense if $y_1(n)$ and $y_2(n)$ were in the same order, so it is necessary to square the output $y_1(n)$. Thus, the statistical dispersion $d(n)$ of input $x(n)$ at time n is given by:

$$d(n) = y_2(n) - {y_1}^2(n) \tag{3.6}$$

where ${y_1}^2(n)$ is defined as the square of sums, and $y_2(n)$ is defined as the sum of squares.

3.2.2 Adaptive Background Modeling Algorithm

The value of dispersion is insignificant when a given input pixel has small variations over time, because the value of ${y_1}^2(n)$ is close to the value of $y_2(n)$. However, the dispersion yields a large value when random changes occur to a given input pixel, because the value of ${y_1}^2(n)$ is significantly different from the value of $y_2(n)$. Based on this observation, the value of dispersion could be used as a threshold to separate the changing foreground pixels from the static background pixels.

As an example, Figure. 3.6, Figure. 3.7, and Figure. 3.8 are the EWMA simulation results along with the statistical dispersion simulation results of a particular pixel on red, green, and blue color channels (in this simulation, $\lambda = 0.1$). The simulation pixel was taken from the video *visiontraffic.avi* in MATLAB [14] vision demos at pixel location $F(i, j) = F(150, 550)$ for every color channel throughout all frames. In Figure. 3.6, Figure. 3.7, and Figure. 3.8, the large values of dispersions at the beginning of the simulation were caused by the initializations of the EWMA filters. Upon the convergences of the filters, filtered outputs were identical and the values of dispersions were zeros. However, the pixel intensities decreased sharply at frame time $n = 320$. This was caused by a moving object entering this pixel location. Furthermore, the dramatic changes in pixel intensities result in significant differences between filtered outputs, thus the values of dispersions were significant.

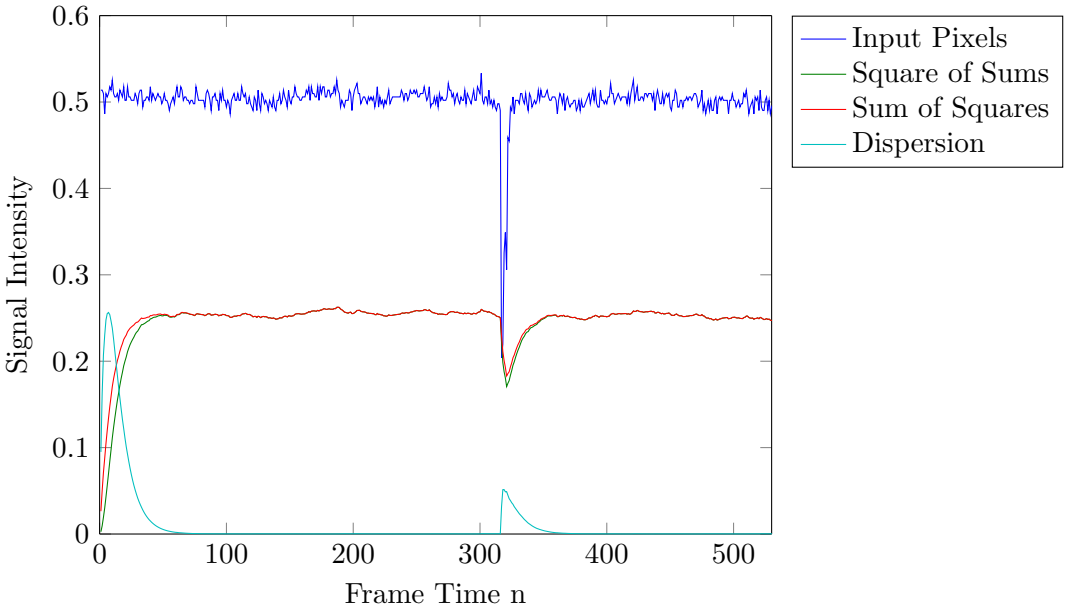


Fig. 3.6: The EWMA filtered dispersion result of a pixel in red color channel.

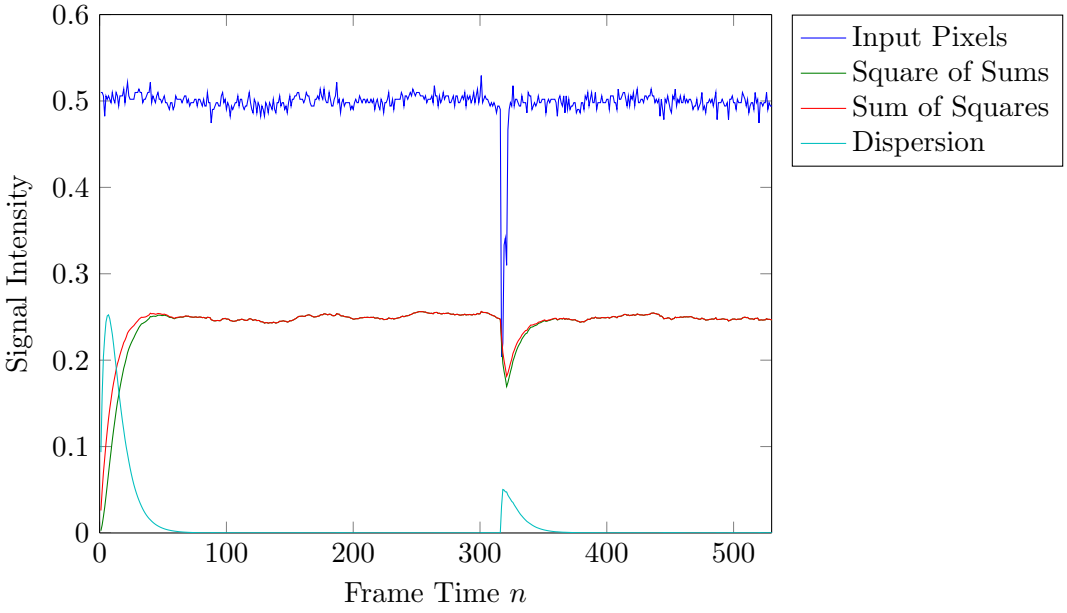


Fig. 3.7: The EWMA filtered dispersion result of a pixel in green color channel.

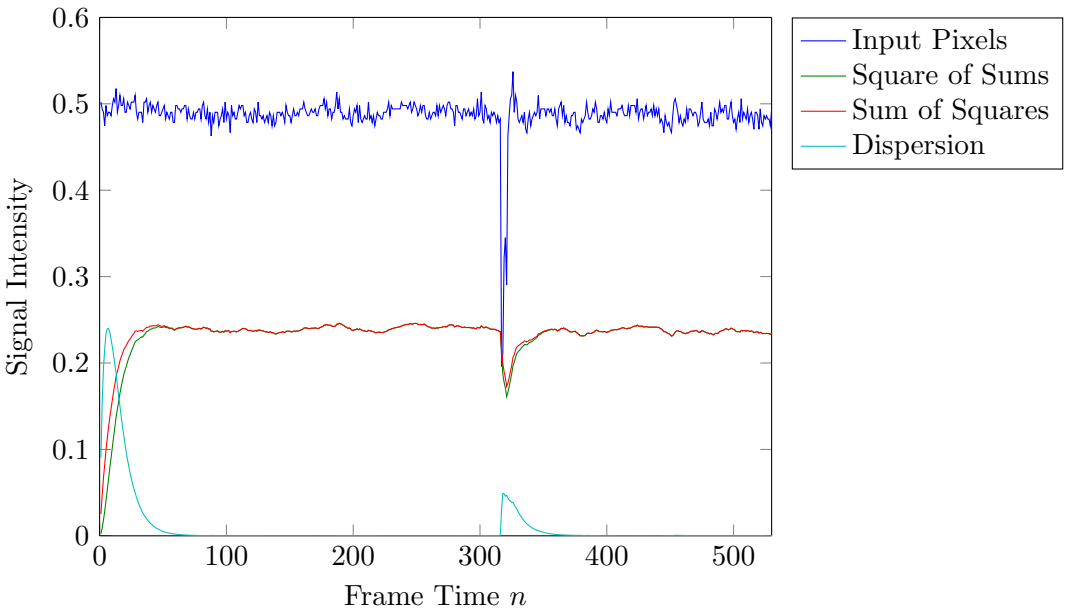


Fig. 3.8: The EWMA filtered dispersion result of a pixel in blue color channel.

At the beginning of the process, $y_1(n-1)$ and $y_2(n-1)$ are both initialized to zero, and it takes some time for the process to converge. This results in value of ${y_1}^2(n)$ being significantly different from the value of $y_2(n)$, and the value of dispersion $d(n)$ being large. Furthermore, $y_1(n)$ and $y_2(n)$ take some time to converge after random changes occurred, because the EWMA filter not only smoothed out the high frequency camera noise, but also smoothed out the fast changes in pixels, this results in the output video sequence being delayed.

In this approach, a system similar to Algorithm. 3.1 is implemented to achieve the goal of adaptive background modeling. First, the system stores past Δ pixel values at past frame time $n-\Delta$ in buffers for each pixel location. Second, the system estimates the statistical dispersion on a pixel-level based on EWMA filtered outputs ${y_1}^2(n)$ and $y_2(n)$. Third, the system uses the cumulative value of statistical dispersion for each color channel as the threshold T to control finite state machines, holding pixel values from the delay buffers, or updating pixel values from current input pixel values. The threshold T is a hand-picked value based on a wide range of tests, $300 \leqslant T \leqslant 600$ worked for most tested video sequences in this research. As a result, the adaptive background model of a given video sequence is

generated from this system.

After some preliminary tests on the implementation of this system, it appears that some pixels in the background models have significant variations. This results in some pixels being held at the initialized values, and theses affected pixels show up on the background models as black dots. Such occurrences were speculated as the drawbacks of pixel-level processing. Based on this assumption, morphological operation was applied to assistance the current implementation, and to improve the quality of background models.

It is obvious that the size of a moving object in a video sequence occupies more than just a single pixel, and the pixels which make up an object tend to have adjacent spatial locations. Based on this characteristic, the nearest neighboring pixels could be segmented as a group of pixels by utilizing morphological operation, specifically dilation. The morphological dilation algorithms are well established in the studies of image processing, therefore morphological dilation algorithm is not presented in this research.

In this modified implementation, a square structural element with a height of h is used for morphological dilation. Instead of holding or updating a single pixel, the system holds a group of pixels when the statistical dispersion is greater than the threshold T and updates a group of pixels when the statistical dispersion is less than the threshold T. The overall design of the modified implementation is given in Algorithm. 3.2. After some initial tests on the modified implementation of statistical dispersion background modeling algorithm, the quality of the background models has been improved.

3.2.3 Notes on Implementation

Similar to the implementation of last approach, the input video sequence was also converted to PPM format before processing to ensure that all the frames are available in the physical memory during run-time. Again, this depends on the length of the input video and the size of the physical memory on the computer, the PPM format may or may not be suggested.

Algorithm 3.2 Statistical Dispersion Background Modeling Algorithm

Input:

Video sequence $F(M, N, K, L)$,
Threshold T,
Frame delay Δ,
Smoothing factor λ,
Height of the structural element h

Output:

Background model video sequence $F_{bg}(I, J, K, Q)$

begin

Initialize dispersions $d(I, J) = 0$
Initialize states of dilation $S(I, J) = 0$
Initialize buffers $F_P(I, J, K, Q) = 0$
Initialize EWMA IIR filters $Y_1(I, J, K) = 0$
Initialize EWMA IIR filters $Y_2(I, J, K) = 0$
for $\{i = 1 : I\}$
  for $\{j = 1 : J\}$
    Reinitialize dispersion $d(i, j) = 0$
    for $\{k = 1 : K\}$
      for $\{n = 1 : Q\}$
        Execute EWMA IIR filters $Y_1(i, j, k)$ with $F(i, j, k, n)$
        Execute EWMA IIR filters $Y_2(i, j, k)$ with $F^2(i, j, k, n)$
        Calculate dispersion $d(i, j)$ with $Y_2(i, j, k)$ and ${Y_1}^2(i, j, k)$
        Update buffers $F_{past}(i, j, k, n)$ from $F(i, j, k, n - \Delta)$
        if $\{d(i, j) >= T\}$
          $S(i, j) = 1$
        end
        if $\{d(i, j) < T\}$
          $S(i, j) = 0$
        end
      end
    end
  end
  if $\{S(i, j) = 1\}$
    Dilate and output background $F_{bg}(i \times h, j \times h, k, n)$ from $F_{past}(i \times h, j \times h, k, n)$
  end
  if $\{S(i, j) = 0\}$
    Dilate and output background $F_{bg}(i \times h, j \times h, k, n)$ from $F(i \times h, j \times h, k, n)$
  end
end

end

3.3 Low Rank and Sparse Based Model

In this section, a novel approach based on low rank and sparsity constraints matrix decomposition is considered to achieve the goal of adaptive background modeling. The challenge of this approach is that it leads to a highly non-convex problem, and there is no closed form solution to this type of problem. Therefore, this problem is solved by relaxing some of the constraints, and sufficient results are obtained by applying greedy methods.

3.3.1 Low Rank and Sparsity Constraints

In this approach, the video sequence is arranged into a large data matrix $M \in \mathbb{R}^{P \times Q}$, and each RGB frame is vectorized into the column of data matrix M. The construction of the data matrix M is illustrated in Figure. 3.9.

Low Rank Constraint for the Background. An important observation is that if the scene is static, then no pixel should change over time. The data matrix M from the video sequence results in the columns of M being identical, thus $rank(M) = 1$. If there are small variations present in the columns of M due to small motions, lighting changes, and camera noises, then M should have a low rank.

Sparsity Constraint for the Foreground. Another important observation is that in the data matrix M from the video sequence, any foreground moving object of interest only occupies a small fraction of the frame. Thus any foreground moving object in the data matrix M tends to be sparse. In contrast, foreground moving objects in the data matrix break previous low rank assumption of the background, therefore the sparse foreground moving objects can be separated from the low rank background scene.

Based on these observations, the data matrix M can be decomposed into foreground component, background component, and noise component. The decomposition of data matrix M can be modeled as follows:

$$M = S + L + N \tag{3.7}$$

where S is the sparse matrix, L is the low rank matrix, and N is the Gaussian noise matrix.

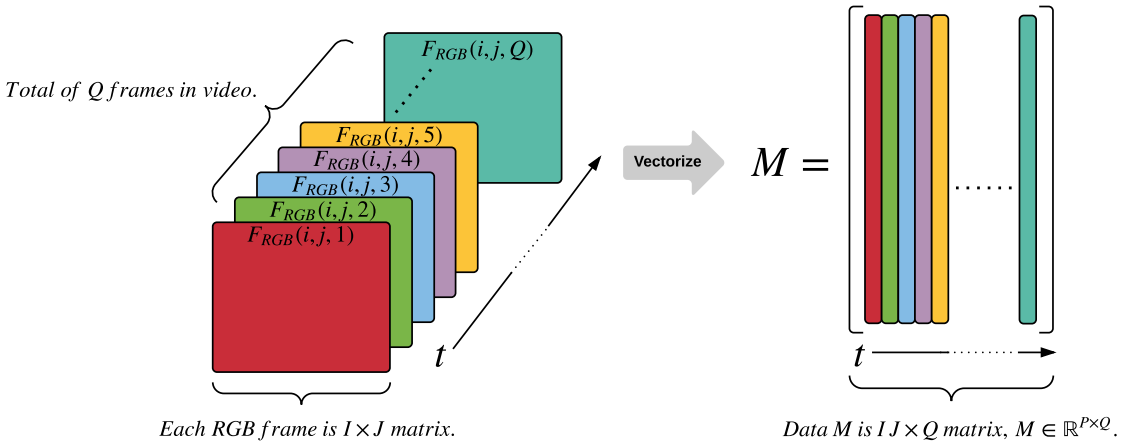


Fig. 3.9: The construction of the data matrix from a video sequence.

By applying the low rank and sparsity constraints, the separation of the foreground moving objects (sparse component) and the background scene (low rank component) can be formulated as following optimization problem:

$$\min_{S,L} \|S\|_0 + \alpha \cdot rank(L) \quad \text{s.t. } \|M - S - L\|_F \leq \varepsilon \tag{3.8}$$

where $\|\cdot\|_0$ is the L_0 norm, and $\|\cdot\|_F$ is the Frobenius norm. The purpose of the L_0 norm is to count the number of non-zero terms of S, and it will ensure that the foreground moving objects are spatially sparse. For further consideration, the objective function of this optimization problem can also be formulated as follows:

$$\min_{S,L} \|S\|_0 + \alpha \cdot rank(L) + \beta \|M - S - L\|_F^2 \tag{3.9}$$

This is not a convex problem, and there exists no closed form solution to this problem. As an alternative, this non-convex problem can be reformulated into a suboptimal optimization problem, and it can be solved iteratively by applying greedy methods. By relaxing some of the constraints, the suboptimal optimization problem can be formulated

as follows:

$$\min_{S,L} \|S\|_0 \quad \text{s.t.} \begin{cases} \|M - S - L\|_F \leq \varepsilon \\ rank(L) = 1 \end{cases} \tag{3.10}$$

$$\min_{S,L} \|S\|_0 + \beta \|M - S - L\|_F^2 \quad \text{s.t. } rank(L) = 1 \tag{3.11}$$

The relaxed low rank constraint $rank(L) = 1$ implies that $L = \underline{l}\,\underline{\mathbf{1}}^\intercal$, thus the suboptimal optimization problem can also be formulated as follows:

$$\min_{S,L} \|S\|_0 + \beta \|M - S - L\|_F^2 \quad \text{s.t. } L = \underline{l}\,\underline{\mathbf{1}}^\intercal \tag{3.12}$$

Furthermore, the relaxation of low rank constraint leads the problem into a simple subproblem of minimizing the rank-1 component:

$$\begin{aligned} \min_{\underline{l}} \|X - \underline{l}\,\underline{\mathbf{1}}^\intercal\|_F^2 &= \operatorname{tr}(X - \underline{l}\,\underline{\mathbf{1}}^\intercal)(X - \underline{l}\,\underline{\mathbf{1}}^\intercal)^\intercal \\ &= \operatorname{tr}(XX^\intercal) - \operatorname{tr}(\underline{l}\,\underline{\mathbf{1}}^\intercal X^\intercal) - \operatorname{tr}(X\underline{l}\,\underline{\mathbf{1}}^\intercal) + \operatorname{tr}(\underline{l}\,\underline{\mathbf{1}}^\intercal \underline{\mathbf{1}}\,\underline{l}^\intercal) \\ &= \|X\|_F^2 - 2\,\underline{l}^\intercal X\,\underline{\mathbf{1}} + Q\,\underline{l}^\intercal \underline{l} \end{aligned}$$

where $X = M - S$, and $\operatorname{tr}(\cdot)$ is the trace.

Now, taking the partial directive respect to $\underline{l}$, set it to zero and solve for $\underline{l}$ to minimize the rank-1 component:

$$\begin{aligned} \frac{\partial}{\partial \underline{l}}(\|X\|_F^2 - 2\,\underline{l}^\intercal X\,\underline{\mathbf{1}} + Q\,\underline{l}^\intercal \underline{l}) &= 0 \\ \implies \quad -2X\underline{\mathbf{1}} + 2Q\underline{l} &= 0 \\ \implies \quad \underline{l} &= \frac{1}{Q} X\,\underline{\mathbf{1}} \end{aligned}$$

An important fact of this subproblem is that $\underline{l} = \frac{1}{Q} X\,\underline{\mathbf{1}}$ is the column average of X, which is also the column average of $M - S$, and recall that $L = \underline{l}\,\underline{\mathbf{1}}^\intercal = \frac{1}{Q} X \underline{\mathbf{1}}\,\underline{\mathbf{1}}^\intercal = \frac{1}{Q}(M - S)\underline{\mathbf{1}}\,\underline{\mathbf{1}}^\intercal$.

Eventually, the low rank constraint of previous problem is reduced, and the suboptimal optimization problem can be formulated as follows:

$$\min_{S} \ \|S\|_0 + \beta\|M - S - \frac{1}{Q}(M - S)\underline{\mathbf{1}}\,\underline{\mathbf{1}}^{\intercal}\|_F^2 \tag{3.13}$$

3.3.2 Adaptive Background Modeling Algorithm

In this approach, a simple algorithm is implemented to solve this highly non-convex problem by applying an iterative greedy method to the reduced suboptimal optimization problem. And the goal of adaptive background modeling is achieved by applying a time windowing algorithm. First, the algorithm initializes $S = 0$, then takes the column average of $M - S$. Second, the algorithm extracts foreground sparse components S, but fill in background low rank components L. Third, the algorithm applies a greedy method to sort out the top 3-5% of S, and then the algorithm iterates. The implementation of the low rank and sparsity constraints background modeling algorithm is given in Algorithm. 3.3.

Algorithm 3.3 Low Rank and Sparsity Constraints Background Modeling Algorithm

Input:

Video sequence $F(I, J, K, Q)$,
Greedy percentage G_{top},
Number of iterations α

Output:

Background video sequence $F_B(I, J, K, Q)$,
Foreground video sequence $F_F(I, J, K, Q)$

begin

Generate data matrix $M_{P,Q}$ from reshaping $F(I, J, K, Q)$
Initialize foreground sparse matrix $S_{P,Q} = 0$
for $\{i = 1 : \alpha\}$
$L_{P,Q} = \frac{1}{Q}(M - S)_{P,Q}\underline{\mathbf{1}}\,\underline{\mathbf{1}}^{\intercal}$
end
$S_{P,Q} = \begin{cases} (M - L)_{P,Q}, & |(M - L)_{P,Q}| \geqslant T \\ 0, & \text{otherwise} \end{cases}$
T chosen to accept 3%$\leq G_{top} \leq$5% of the largest values in $S_{P,Q}$
Output $F_B(I, J, K, Q)$ from reshaping $L_{P,Q}$
Output $F_F(I, J, K, Q)$ from reshaping $S_{P,Q}$

end

After some preliminary tests, the low rank and sparsity constraints background modeling algorithm produced some very useful results. Furthermore, this algorithm converges rapidly, and only 2-3 iterations are needed. However, it failed to update the temporal features in a video sequence, because this algorithm requires all the frames in a video sequence to separate the foreground moving objects and background scenes. Due to the needs of updating temporal features in this research, a system employs a sliding window algorithm and the low rank and sparsity constraints background modeling algorithm is implemented.

Instead processing all Q frames at a time, the sliding window low rank and sparsity constraints background modeling algorithm processes W frames at a time. The data matrix M now has dimension of $P \times W$, $M \in \mathbb{R}^{P \times W}$. At the beginning, the system processes the first W columns in the data matrix. The system then slides the window through time to get a new frame, re-initializes the column which corresponds to the new frame in S to be zeros, and processes the new block of data using the same algorithm. The idea of the sliding window process is illustrated in Figure. 3.10, and the implementation of the sliding window combined with low rank and sparsity constraints background modeling algorithm is given in Algorithm. 3.4. The size of sliding window W is a hand-picked value based on a wide range of tests, and it is typically greater than the frame rate FPS of a given video sequence. A good range of the sliding window size is $\text{FPS} \leqslant W \leqslant 2 \cdot \text{FPS}$.

3.3.3 Notes on Implementation

The proposed algorithm only requires some inexpensive mathematical operations while some previous robust principal component analysis (RPCA) algorithms require expensive mathematical operations to accomplish the same goal. For instance, RPCA via principal component pursuit (RPCA-PCP) proposed by Candes et al. [11] and RPCA via proximal gradient (RPCA-PG) proposed by Wright et al. [12] both require expensive singular value decomposition (SVD) operations to produce singular value thresholding operators, thus this type of approach is hard to implement in real-time. However, the proposed algorithm applies just mean and sort operations to produce useful results, and it is very easy to implement for real-time applications.

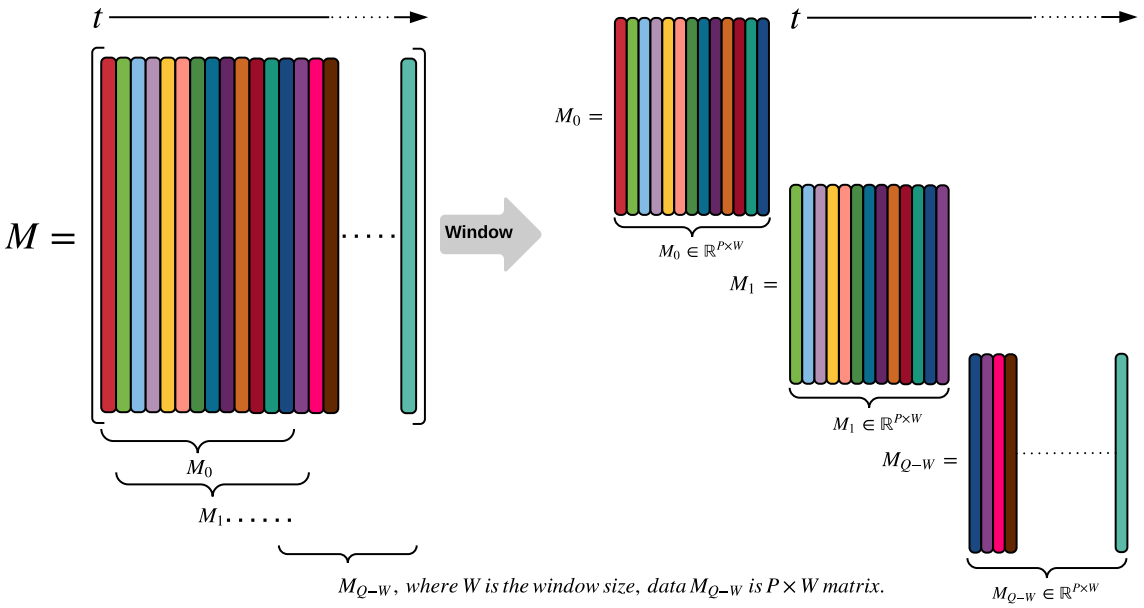


Fig. 3.10: The illustration of the sliding window process.

Algorithm 3.4 Sliding Window Algorithm

Input:

Video sequence $F(I, J, K, Q)$,
Size of sliding window W

Output:

Background video sequence $F_B(I, J, K, Q)$
Foreground video sequence $F_F(I, J, K, Q)$

begin

Generate the first data window $M_{P,W}$ from reshaping $F(I, J, K, Q)$
Initialize foreground sparse matrix $S_{P,W} = 0$
Process the first data window $M_{P,W}$ with Algorithm. 3.3
Output $F_B(I, J, K, W)$ from reshaping $L_{P,W}$.
Output $F_F(I, J, K, W)$ from reshaping $S_{P,W}$.
while {has frame}
Update the last column of $M_{P,W}$ from reshaping $F(I, J, K, Q)$
Make copy of sparse matrix $S_{P,W}$ from previous $S_{P,W}$
Re-initialize the last column of $S_{P,W} = 0$
Process updated $M_{P,W}$ and $S_{P,W}$ with Algorithm. 3.3
Output $F_B(I, J, K, W)$ from reshaping $L_{P,W}$.
Output $F_F(I, J, K, W)$ from reshaping $S_{P,W}$.
end

end

CHAPTER 4

Results and Comparisons

In this chapter, three adaptive background models are tested on six video sequences for the subjective evaluations. The experimental data sets are described in the first section, the qualitative results are presented and some issues with the experiments and results are discussed in the second section.

4.1 Experimental Data

Six video sequences with different characteristics were chosen from three different data sources for the experiments. In this section, a brief description of each scenario and the characteristics of each video sequence are presented.

4.1.1 MATLAB Data

MATLAB toolbox vision data set [14] consists of twenty six video sequences, and two video sequences are chosen because of how frequent these videos were used in the field of background modeling and background subtraction. In this data set, each frame has a size of 640×360 with three color channels and a frame rate of 30 FPS.

- **Vision Traffic**: Vehicles drive past a highway surveillance camera in different lanes with different speeds. This video sequence consists of 531 RGB frames.
- **Atrium**: Several people walk past an atrium in different directions, and one person sets down a white coffee cup on a table located in the atrium. This video sequence consists of 1410 RGB frames.

4.1.2 Xiaoli Li's Data

Xiaoli Li's data set [15] consists of nine video sequences, and two video sequences were chosen because of the presence of dynamic backgrounds or illumination changes. In the first

video, each frame has a size of 160×128 with three color channels. In the second video, each frame has a size of 320×256 with three color channels. Both video sequences have a frame rate of 30 FPS.

- **Switch Light**: Several people walk through a room with a changing level of background illumination. This video sequence consists of 1546 RGB frames.
- **Shopping Mall**: A shopping mall with a crowded scene of people consistently appearing in each frame. This video sequence consists of 1286 RGB frames.

4.1.3 Lab Data

Lab data set consists of twelve video sequences, and two video sequences were chosen to present the performance of current lab configuration. In this data set, each frame has a size of 800×600 with three color channels and a frame rate of 30 FPS.

- **Backpack**: A student walks through a room, and sets down a backpack. Then the student reenters the room, picks up the backpack, and exits the scene. This video sequence consists of 575 RGB frames.
- **Lecture**: A professor writes on a whiteboard, and exits the scene. Then the professor reenters the scene, continues writing, and exits the scene again. This video sequence consists of 1286 RGB frames.

4.2 Experimental Results

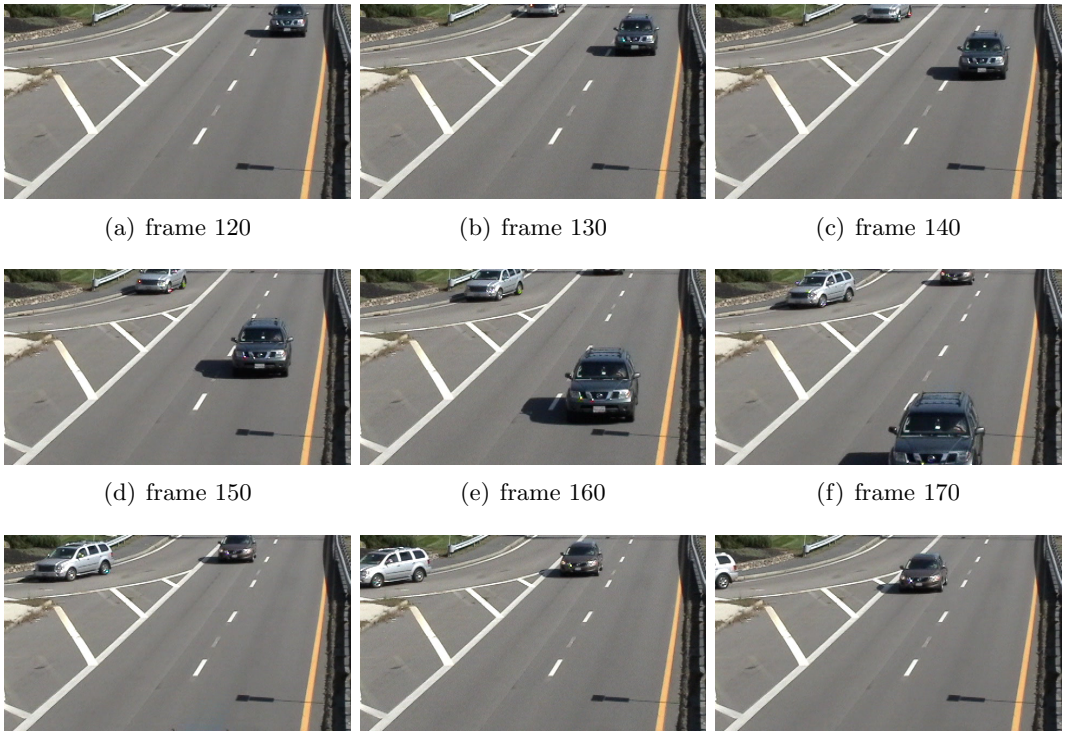
In this section, the qualitative results of six video sequences are presented for the subjective evaluation of three adaptive background models. For the purpose of showing moving objects progressing through each video sequence, nine frames were selected for each figure to illustrate the results, and each selected frame is 10 frames away from the next selected frame. For each video sequence, nine frames from the original video sequence are provided, as well as nine frames from the same video sequence processed with each background modeling algorithm.

4.2.1 Vision Traffic Results

For the video "vision traffic", background models were successfully produced by all algorithms. There are some "ghost" artifacts in the background video produced by adaptive linear prediction based model. This is due to the drawbacks of the pixel-level processing and time delay of the updates. There are less "ghost" artifacts in the background video produced by statistical dispersion based model. In the low rank and sparse based model, the "ghost" artifacts of the moving objects are not visible in the background video. Figure 4.1 is the original frames from the video sequence, Figure 4.2 is the adaptive linear prediction background models of the video sequence, Figure 4.3 is the statistical dispersion background models of the video sequence, Figure 4.4 is the low rank and sparse background models of the video sequence, and Figure 4.5 is the low rank and sparse foreground models of the video sequence.

4.2.2 Atrium Results

For the video "atrium", all three algorithms produced acceptable background models, and successfully updated the appearance of the coffee cup. Similar to the results from the last video, the "ghost" artifacts also exists in the adaptive linear prediction based model and the statistical dispersion based model, but the "ghost" artifacts do not exist in the low rank and sparse based model. The statistical dispersion based model failed to update a small region of the background pixels, because there are always objects moving through the same location in the frames. Figure 4.6 is the original frames from the video sequence, Figure 4.7 is the adaptive linear prediction background models of the video sequence, Figure 4.8 is the statistical dispersion background models of the video sequence, Figure 4.9 is the low rank and sparse background models of the video sequence, and Figure 4.10 is the low rank and sparse foreground models of the video sequence.

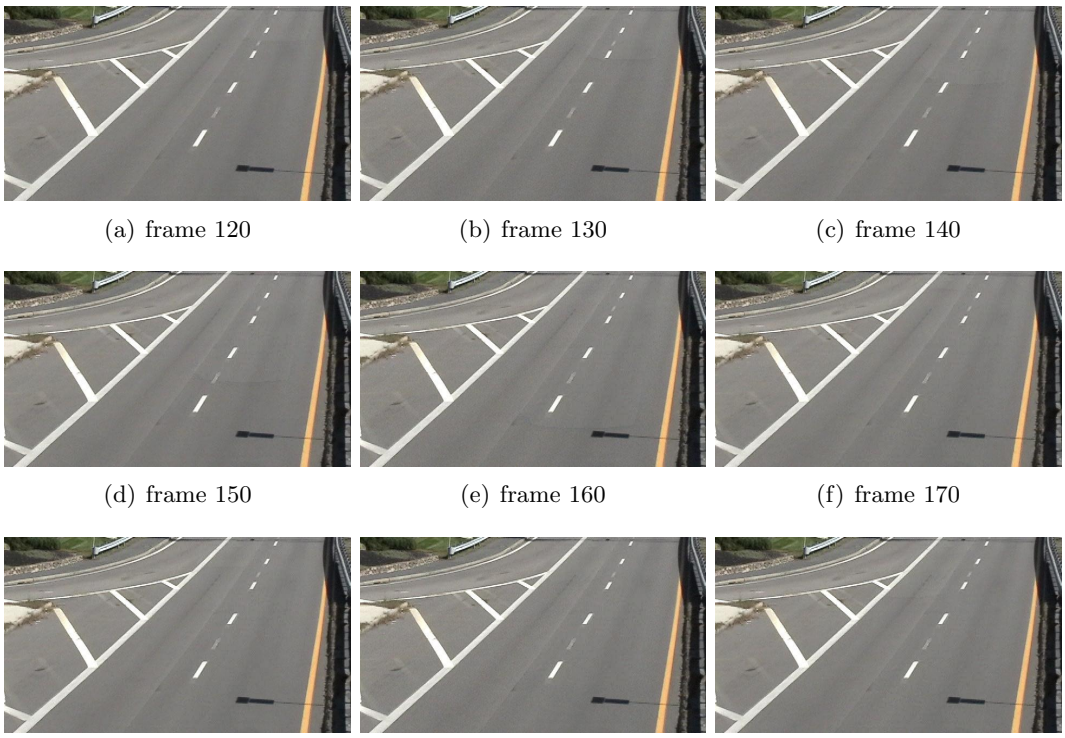


(a) frame 120 (b) frame 130 (c) frame 140

(d) frame 150 (e) frame 160 (f) frame 170

(g) frame 180 (h) frame 190 (i) frame 200

Fig. 4.1: Original frames from video sequence *visiontraffic.avi*.

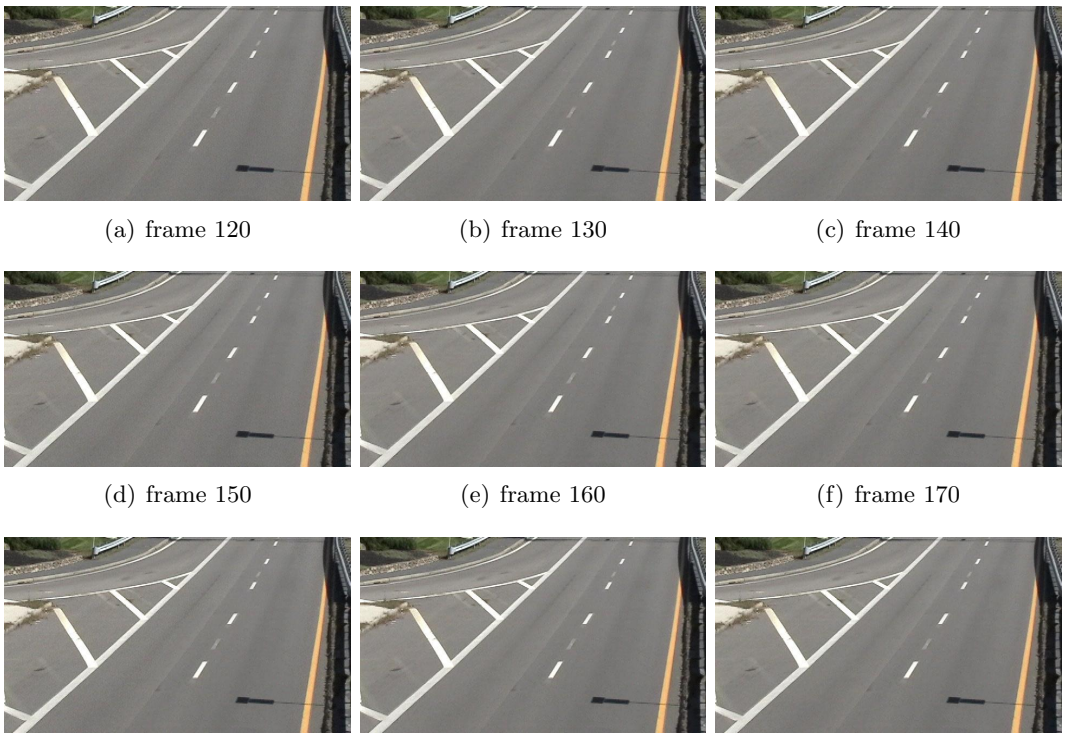


(a) frame 120 (b) frame 130 (c) frame 140

(d) frame 150 (e) frame 160 (f) frame 170

(g) frame 180 (h) frame 190 (i) frame 200

Fig. 4.2: Adaptive linear prediction background models of video sequence *visiontraffic.avi*.

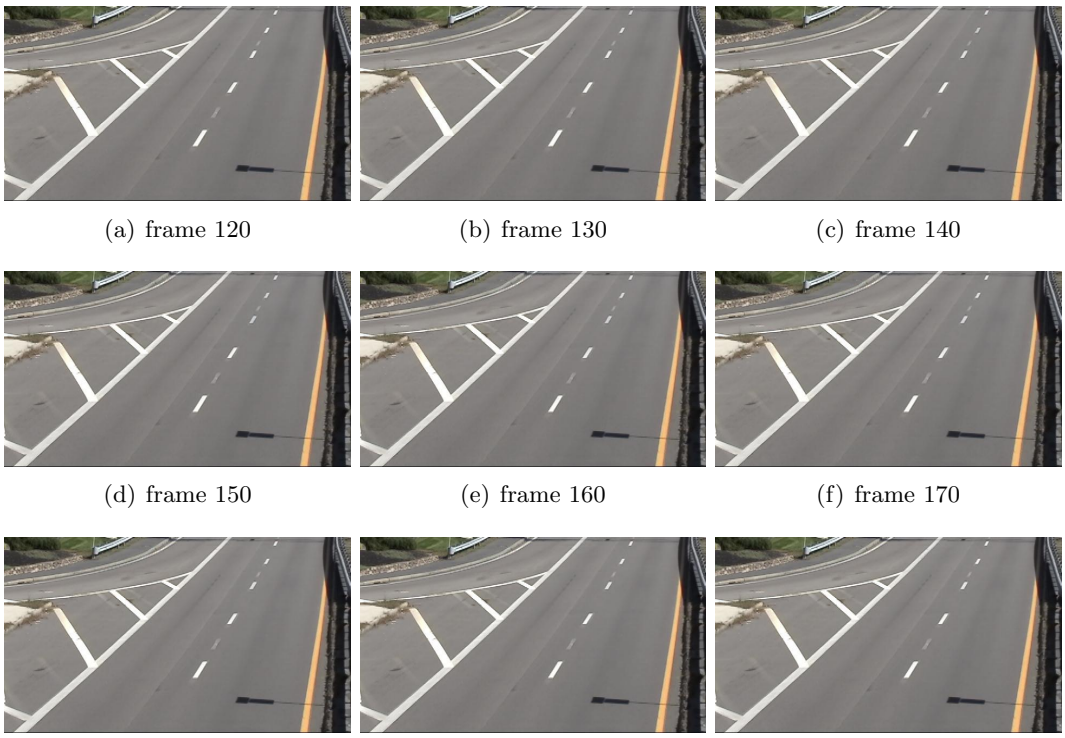


(a) frame 120 (b) frame 130 (c) frame 140

(d) frame 150 (e) frame 160 (f) frame 170

(g) frame 180 (h) frame 190 (i) frame 200

Fig. 4.3: Statistical dispersion background models of video sequence *visiontraffic.avi*.

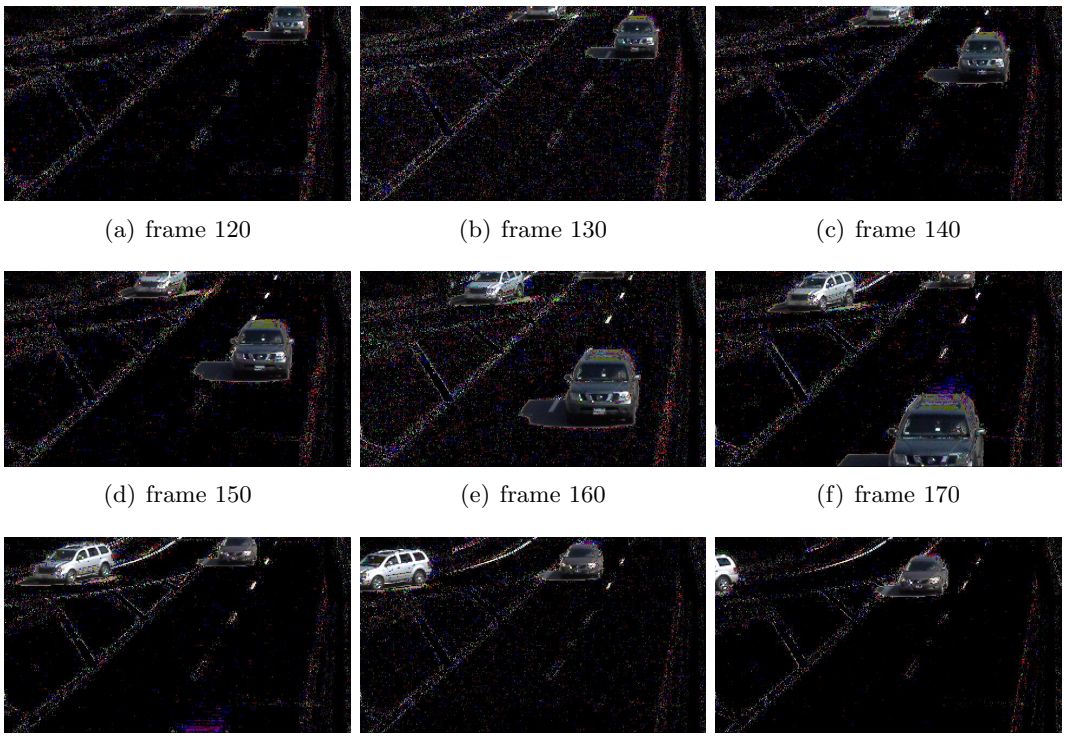


(a) frame 120 (b) frame 130 (c) frame 140

(d) frame 150 (e) frame 160 (f) frame 170

(g) frame 180 (h) frame 190 (i) frame 200

Fig. 4.4: Low rank and sparse background models of video sequence *visiontraffic.avi*.

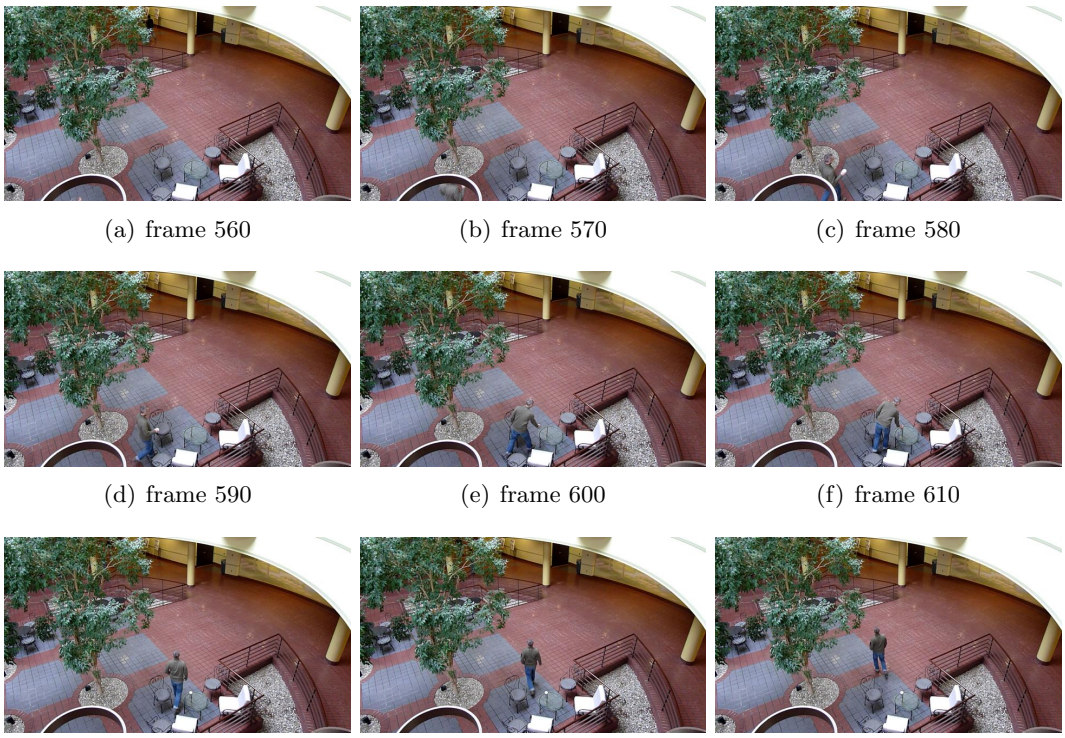


(a) frame 120 (b) frame 130 (c) frame 140

(d) frame 150 (e) frame 160 (f) frame 170

(g) frame 180 (h) frame 190 (i) frame 200

Fig. 4.5: Low rank and sparse foreground models of video sequence *visiontraffic.avi*.

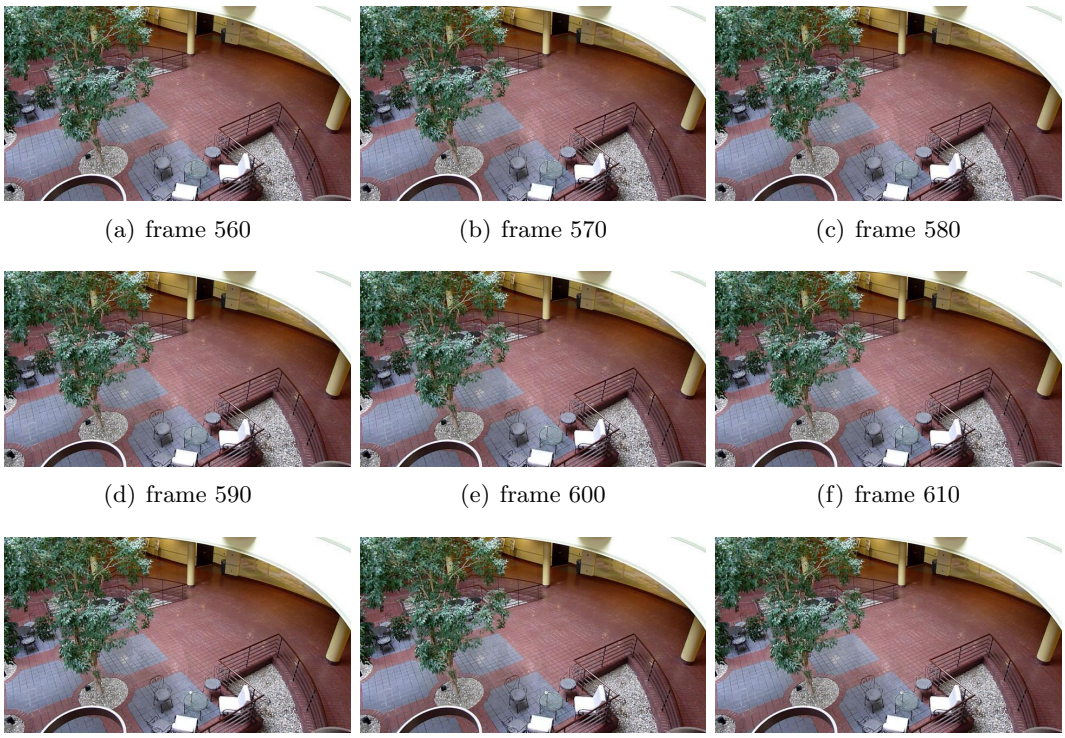


(a) frame 560 (b) frame 570 (c) frame 580

(d) frame 590 (e) frame 600 (f) frame 610

(g) frame 620 (h) frame 630 (i) frame 640

Fig. 4.6: Original frames of video sequence *atrium.avi*.

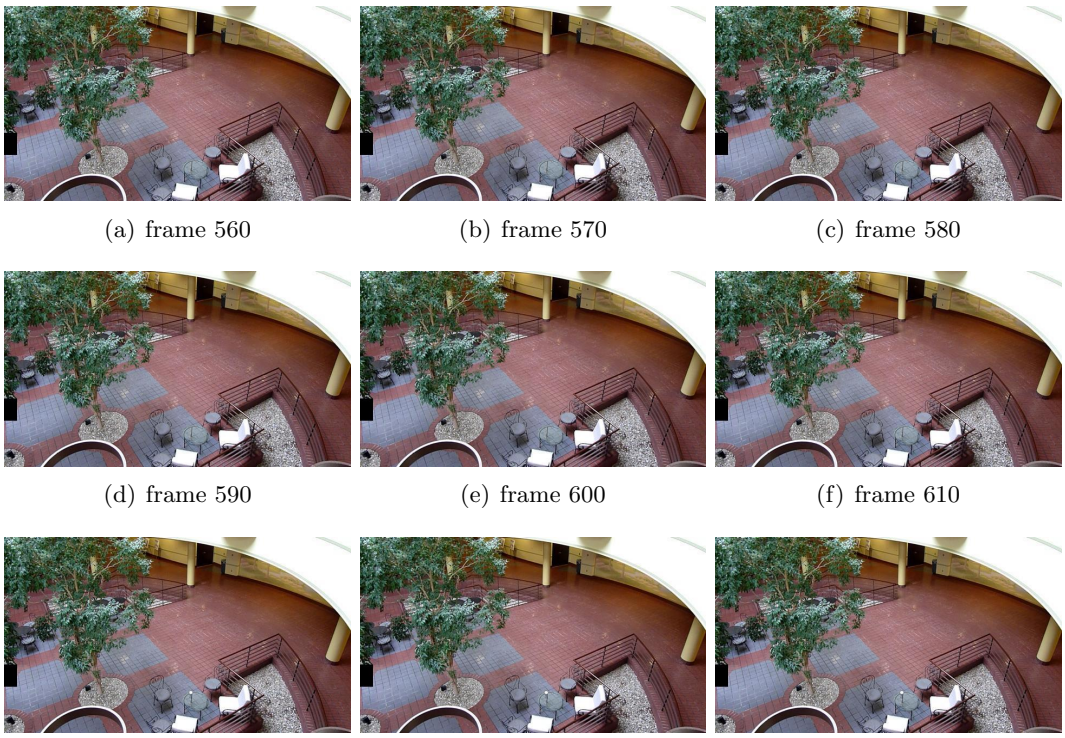


(a) frame 560 (b) frame 570 (c) frame 580

(d) frame 590 (e) frame 600 (f) frame 610

(g) frame 620 (h) frame 630 (i) frame 640

Fig. 4.7: Adaptive linear prediction background models of video sequence *atrium.avi*.

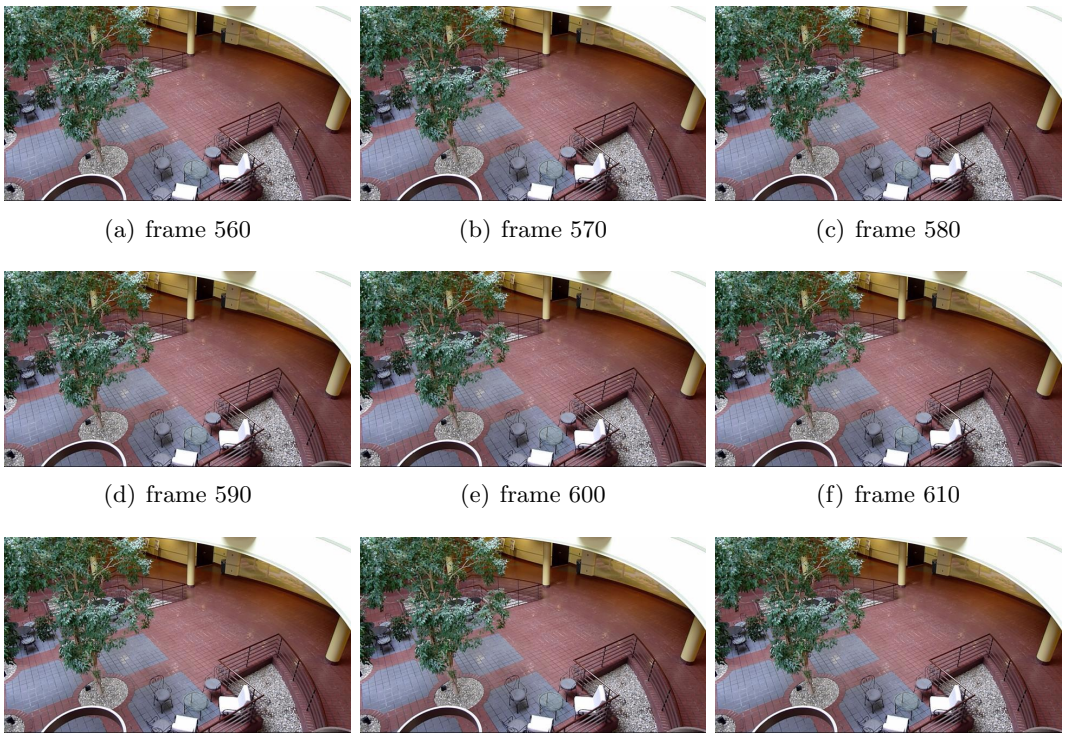


(a) frame 560 (b) frame 570 (c) frame 580

(d) frame 590 (e) frame 600 (f) frame 610

(g) frame 620 (h) frame 630 (i) frame 640

Fig. 4.8: Statistical dispersion background models of video sequence *atrium.avi*.

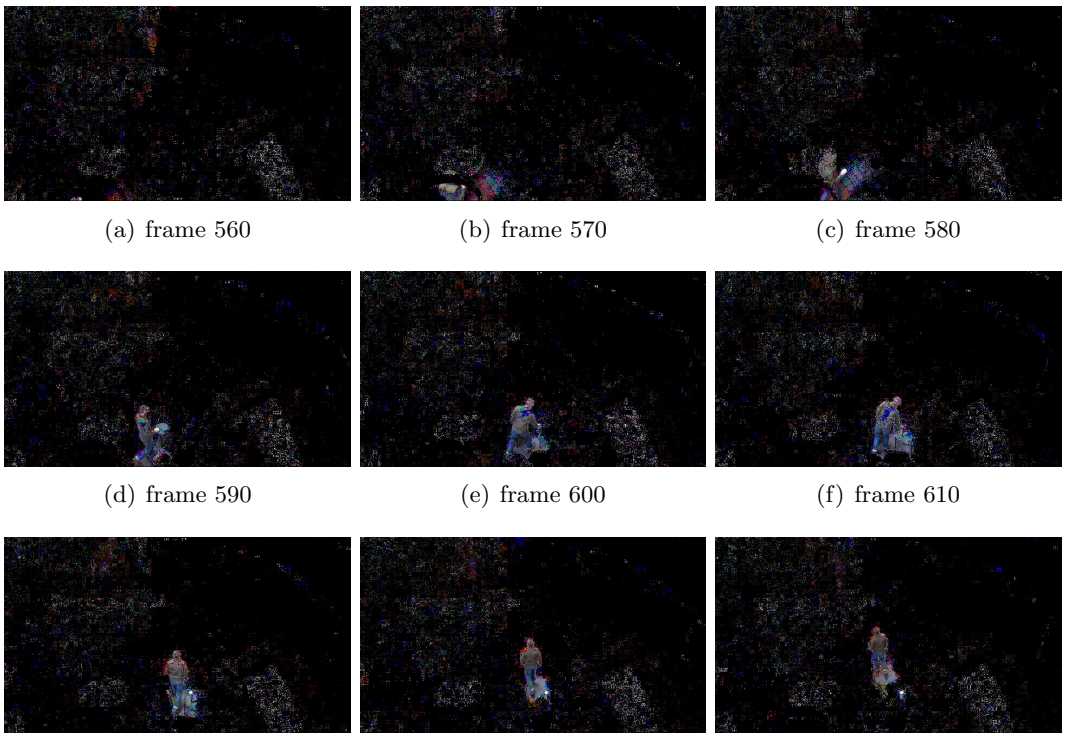


(a) frame 560 (b) frame 570 (c) frame 580

(d) frame 590 (e) frame 600 (f) frame 610

(g) frame 620 (h) frame 630 (i) frame 640

Fig. 4.9: Low rank and sparse background models of video sequence *atrium.avi*.



(a) frame 560 (b) frame 570 (c) frame 580

(d) frame 590 (e) frame 600 (f) frame 610

(g) frame 620 (h) frame 630 (i) frame 640

Fig. 4.10: Low rank and sparse foreground models of video sequence *atrium.avi*.

4.2.3 Switch Light Results

For the video "switch light", all three algorithms produced acceptable background models, and updated the dynamic background caused by the illumination changes. However, for the adaptive linear prediction based model, the path of the moving object is illuminated when the lights are switched off, thus producing dramatic "ghost" artifacts. The frame size of this video is relatively small compared to the structural element used in the statistical dispersion based model. This results in the transition from the initialization to the desired background being rough. The low rank and sparse based model produced an ideal dynamic background model, and the adaptations to the illumination changes are smooth and effective. Figure 4.11 is the original frames from the video sequence, Figure 4.12 is the adaptive linear prediction background models of the video sequence, Figure 4.13 is the statistical dispersion background models of the video sequence, Figure 4.14 is the low rank and sparse background models of the video sequence, and Figure 4.15 is the low rank and sparse foreground models of the video sequence.

4.2.4 Shopping Mall Results

For the video "shopping mall", some of the objects stop moving and become a part of the background, and after a short amount of time these objects started to move again. This results in those objects with uncertain motions fading in and out in all three background models. The adaptive linear prediction based model and the low rank and sparse based model produced acceptable results for such a complex background. However, the statistical dispersion based model failed to model the background on the upper right corner of the video sequence. Because an object is in constant motion at that location, the statistical dispersion background modeling algorithm fails to update the background, and holds that region of pixels at the initialization. Figure 4.16 is the original frames from the video sequence, Figure 4.17 is the adaptive linear prediction background models of the video sequence, Figure 4.18 is the statistical dispersion background models of the video sequence, Figure 4.19 is the low rank and sparse background models of the video sequence, and Figure 4.20 is the low rank and sparse foreground models of the video sequence.

(a) frame 360 (b) frame 370 (c) frame 380

(d) frame 390 (e) frame 400 (f) frame 410

(g) frame 420 (h) frame 430 (i) frame 440

Fig. 4.11: Original frames of video sequence *switchlight.avi*.

(a) frame 360 (b) frame 370 (c) frame 380

(d) frame 390 (e) frame 400 (f) frame 410

(g) frame 420 (h) frame 430 (i) frame 440

Fig. 4.12: Adaptive linear prediction background models of video sequence *switchlight.avi*.

(a) frame 360 (b) frame 370 (c) frame 380

(d) frame 390 (e) frame 400 (f) frame 410

(g) frame 420 (h) frame 430 (i) frame 440

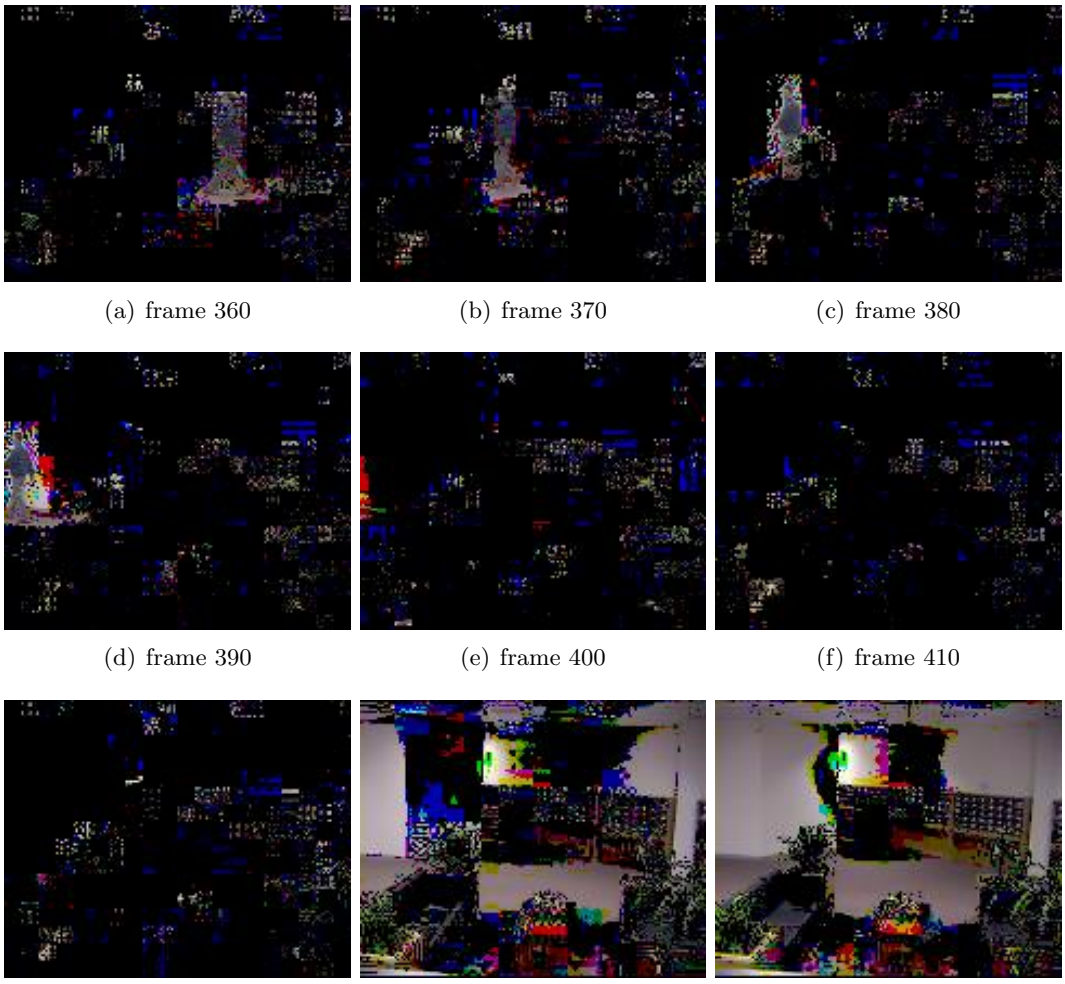
Fig. 4.13: Statistical dispersion background models of video sequence *switchlight.avi*.

(a) frame 360 (b) frame 370 (c) frame 380

(d) frame 390 (e) frame 400 (f) frame 410

(g) frame 420 (h) frame 430 (i) frame 440

Fig. 4.14: Low rank and sparse background models of video sequence *switchlight.avi*.



(a) frame 360 (b) frame 370 (c) frame 380

(d) frame 390 (e) frame 400 (f) frame 410

(g) frame 420 (h) frame 430 (i) frame 440

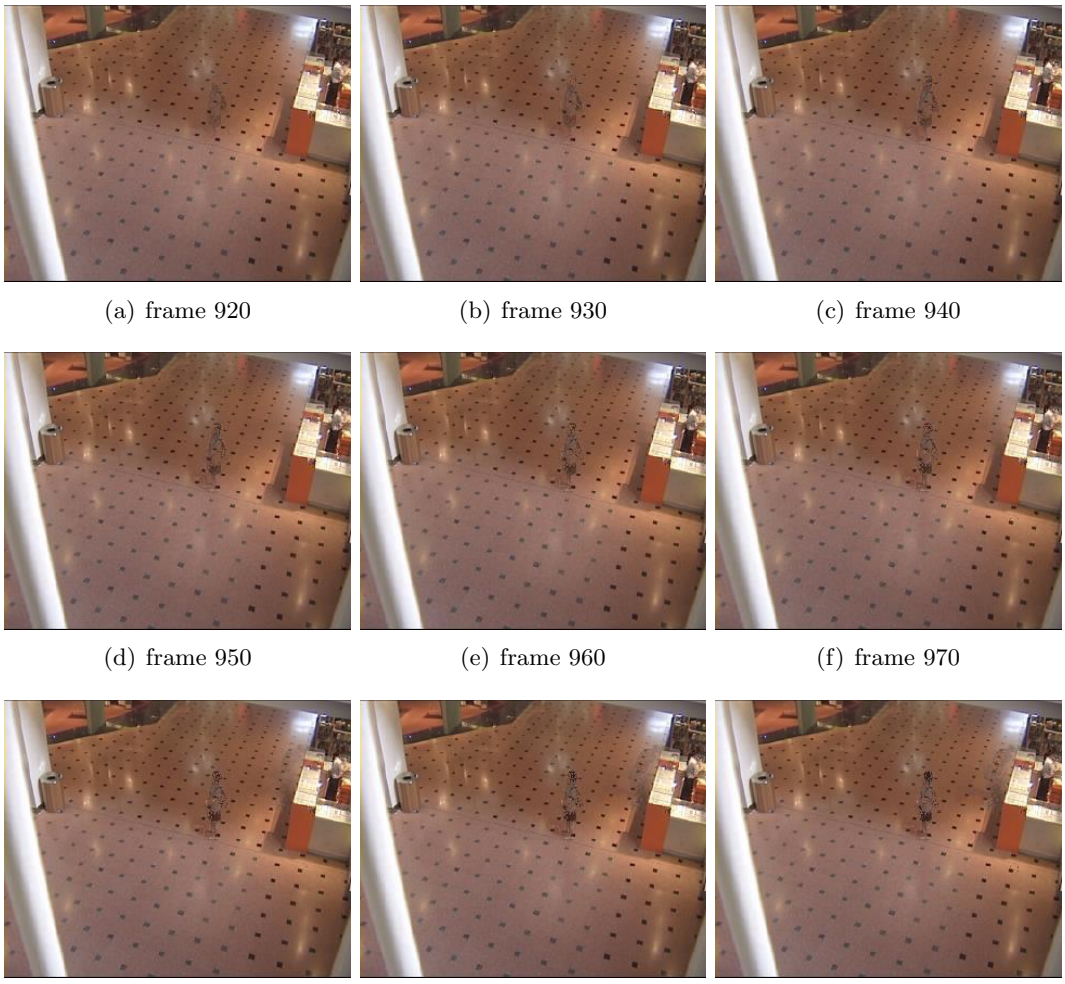
Fig. 4.15: Low rank and sparse foreground models of video sequence *switchlight.avi*.

(a) frame 920 (b) frame 930 (c) frame 940

(d) frame 950 (e) frame 960 (f) frame 970

(g) frame 980 (h) frame 990 (i) frame 1000

Fig. 4.16: Original frames of video sequence *shoppingmall.avi*.

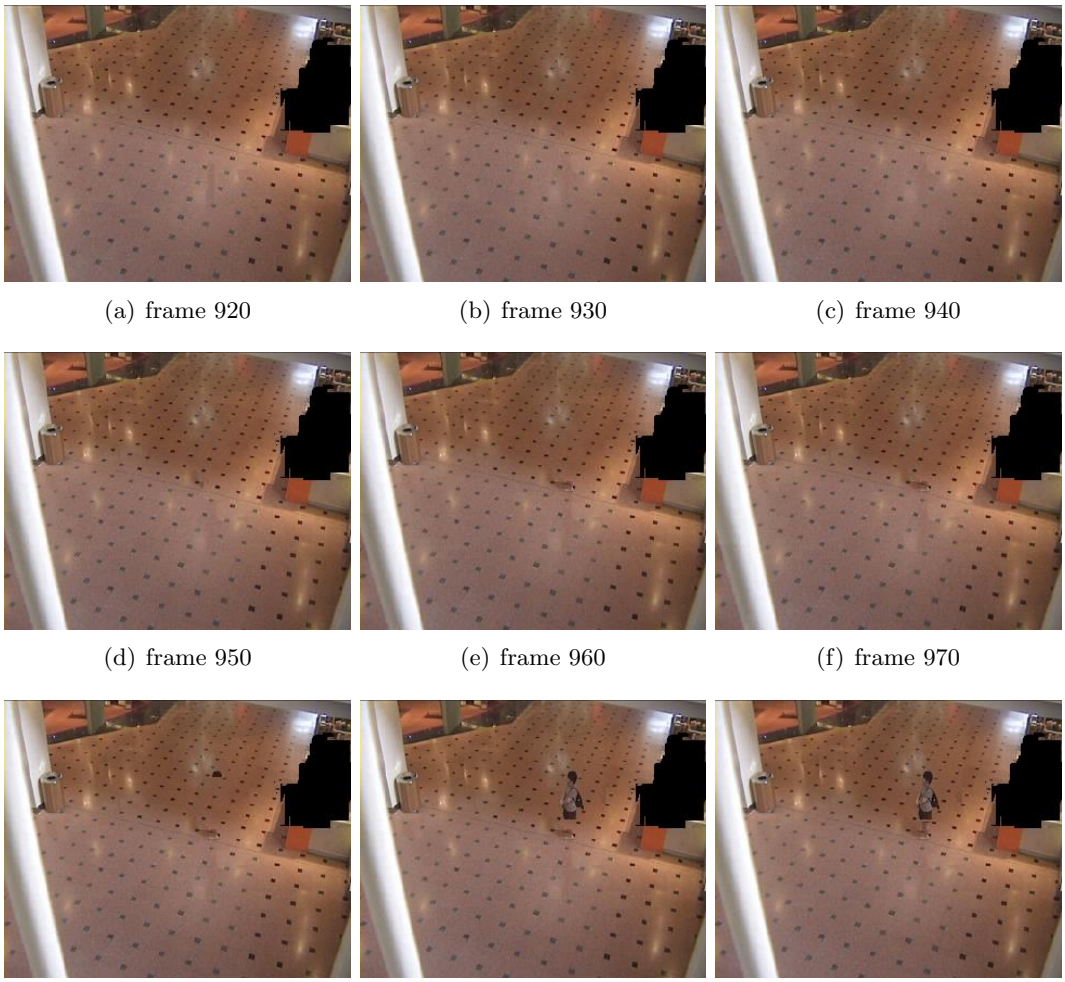


(a) frame 920 (b) frame 930 (c) frame 940

(d) frame 950 (e) frame 960 (f) frame 970

(g) frame 980 (h) frame 990 (i) frame 1000

Fig. 4.17: Adaptive linear prediction background models of video sequence *shoppingmall.avi*.

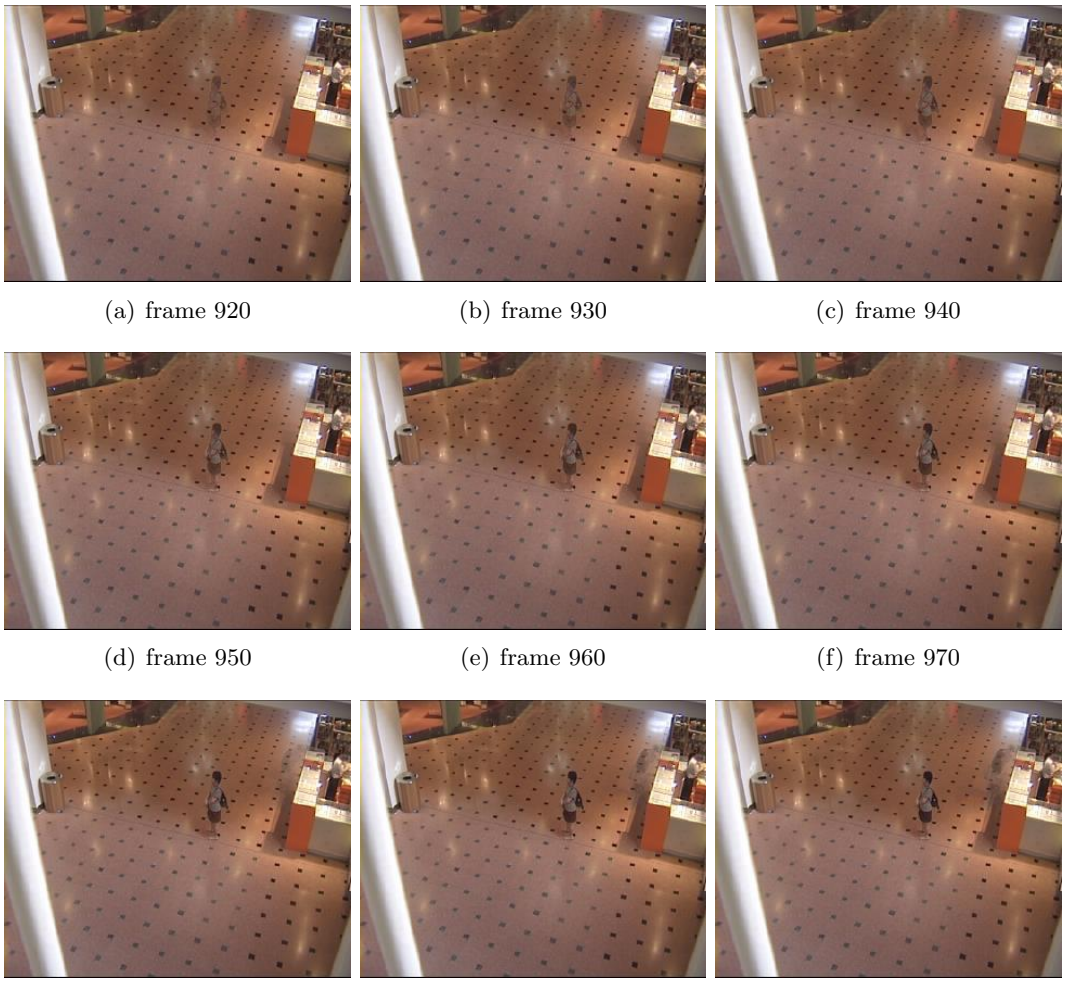


(a) frame 920 (b) frame 930 (c) frame 940

(d) frame 950 (e) frame 960 (f) frame 970

(g) frame 980 (h) frame 990 (i) frame 1000

Fig. 4.18: Statistical dispersion background models of video sequence *shoppingmall.avi*.

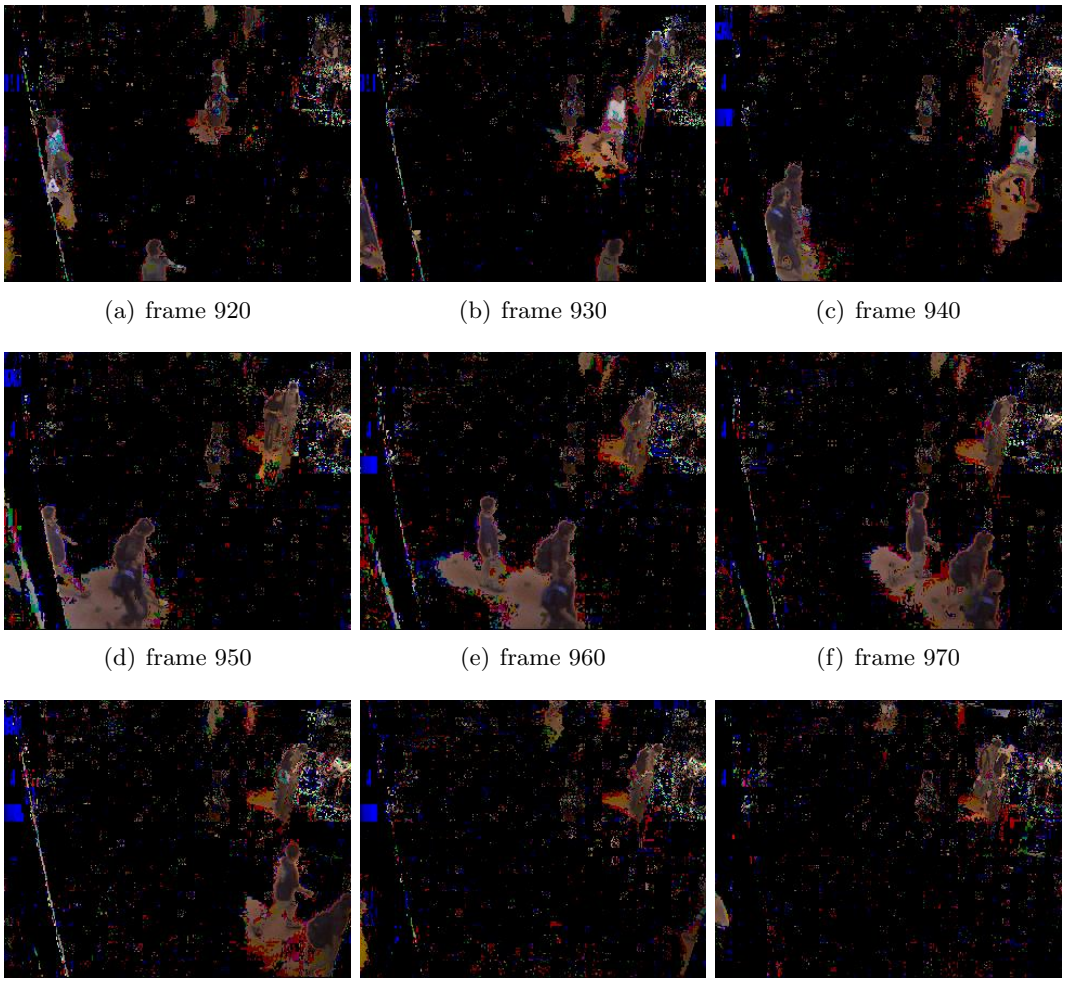


(a) frame 920 (b) frame 930 (c) frame 940

(d) frame 950 (e) frame 960 (f) frame 970

(g) frame 980 (h) frame 990 (i) frame 1000

Fig. 4.19: Low rank and sparse background models of video sequence *shoppingmall.avi*.



(a) frame 920 (b) frame 930 (c) frame 940

(d) frame 950 (e) frame 960 (f) frame 970

(g) frame 980 (h) frame 990 (i) frame 1000

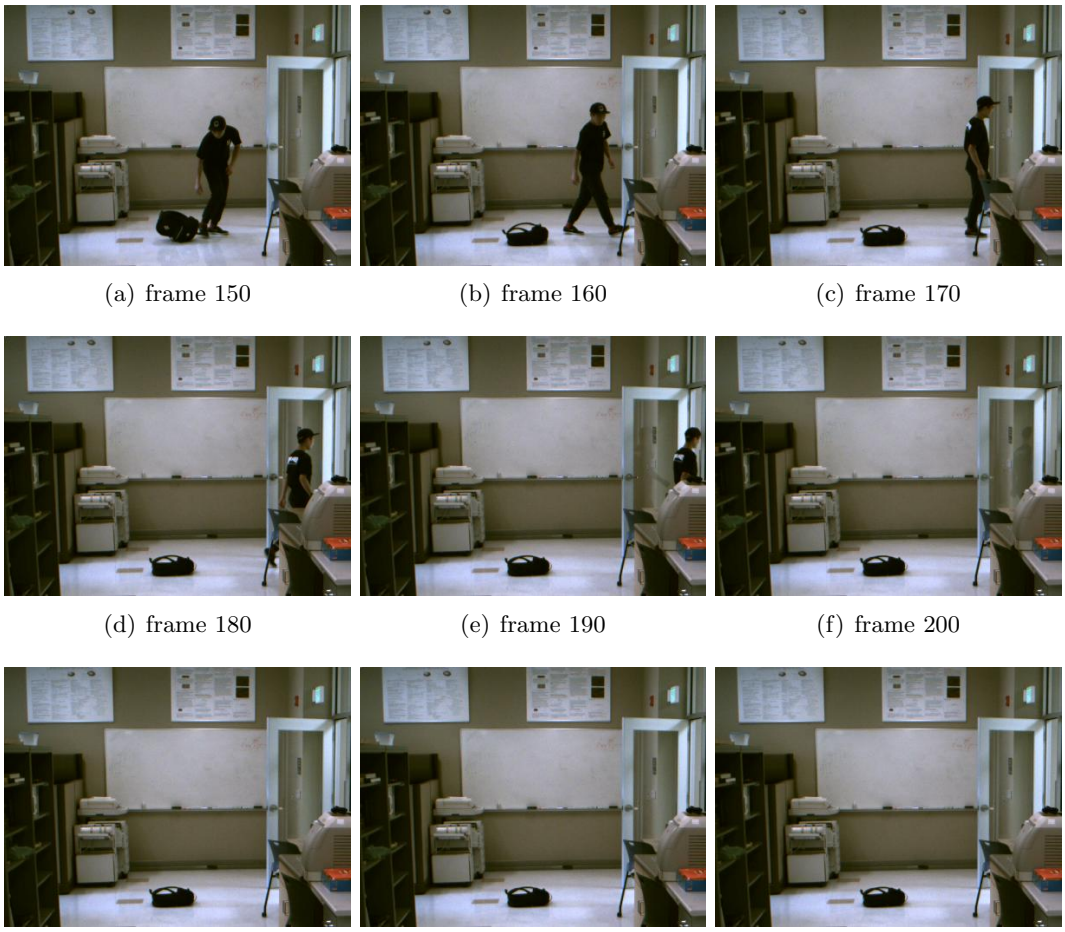
Fig. 4.20: Low rank and sparse foreground models of video sequence *shoppingmall.avi*.

4.2.5 Backpack Results

For the video "backpack", all three algorithms successfully produced background models, and effectively updated the appearance of the backpack. Because the pixels in the background are brighter than previous video sequences, the adaptive linear prediction based model has less "ghost" artifacts compared to previous results, but the backpack does not appear completely due to drawbacks of the pixel-level processing. However, the statistical dispersion based model updated the backpack clearly and completely, and the "ghost" artifacts are nearly absent. The low rank and sparse based model is more sensitive to the temporal motion of the moving objects. This results in the moving objects appearing as part of the background more frequently. Figure 4.21 is the original frames from the video sequence, Figure 4.22 is the adaptive linear prediction background models of the video sequence, Figure 4.23 is the statistical dispersion background models of the video sequence, Figure 4.24 is the low rank and sparse background models of the video sequence, and Figure 4.25 is the low rank and sparse foreground models of the video sequence.

4.2.6 Lecture Results

For the video "lecture", the adaptive linear prediction based model failed to produce a satisfactory background model, and failed to update the notes on the white board, because the notes on the whiteboard are very precise on a pixel-level. A desirable background model was obtained by the statistical dispersion based model, and it effectively updated the notes on the whiteboard. However, the moving object and the whiteboard in the background have a nearly indistinguishable color. This results in some visible "ghost" artifacts being present throughout the background video sequence, but notes on the whiteboard are not obscured. For the low rank and sparse based model, in order to separate the moving object from the background, the size of the moving object should be relatively smaller than the size of the background. This video senescence does not satisfy this condition, thus making separation difficult. However, low rank and sparse based model still produces useful results, because the note on the whiteboard are still visible and distinguishable from the moving object. Figure 4.26 is the original frames from the video sequence, Figure 4.27 is

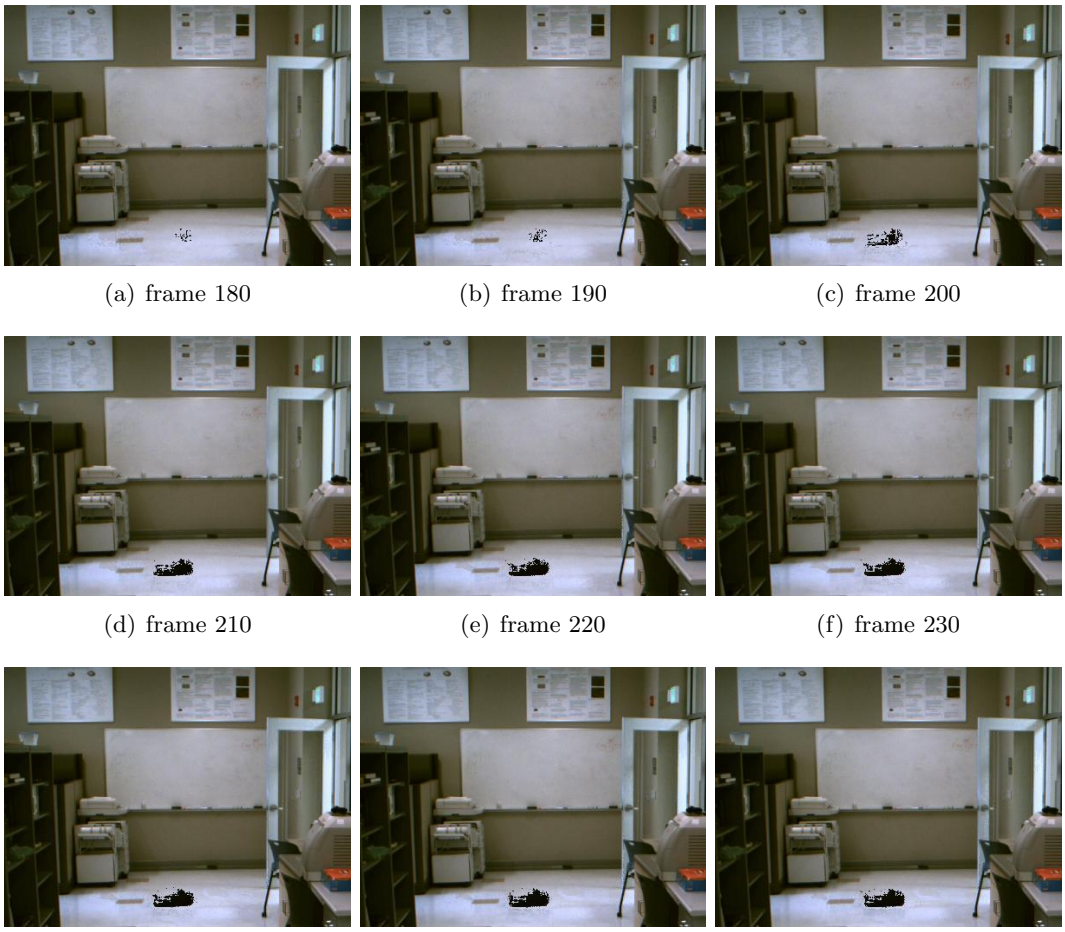


(a) frame 150 (b) frame 160 (c) frame 170

(d) frame 180 (e) frame 190 (f) frame 200

(g) frame 210 (h) frame 220 (i) frame 230

Fig. 4.21: Original frames of video sequence *backpack.avi*.

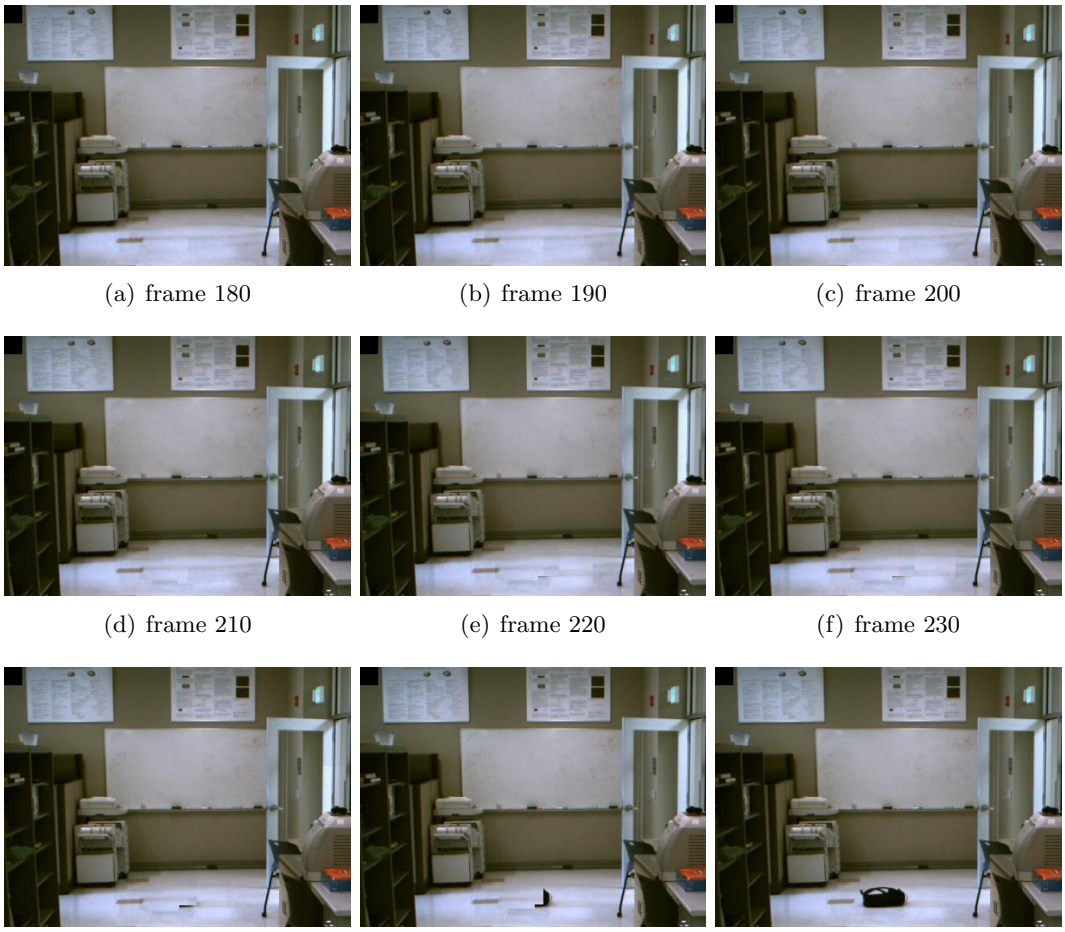


(a) frame 180 (b) frame 190 (c) frame 200

(d) frame 210 (e) frame 220 (f) frame 230

(g) frame 240 (h) frame 250 (i) frame 260

Fig. 4.22: Adaptive linear prediction background models of video sequence *backpack.avi*.

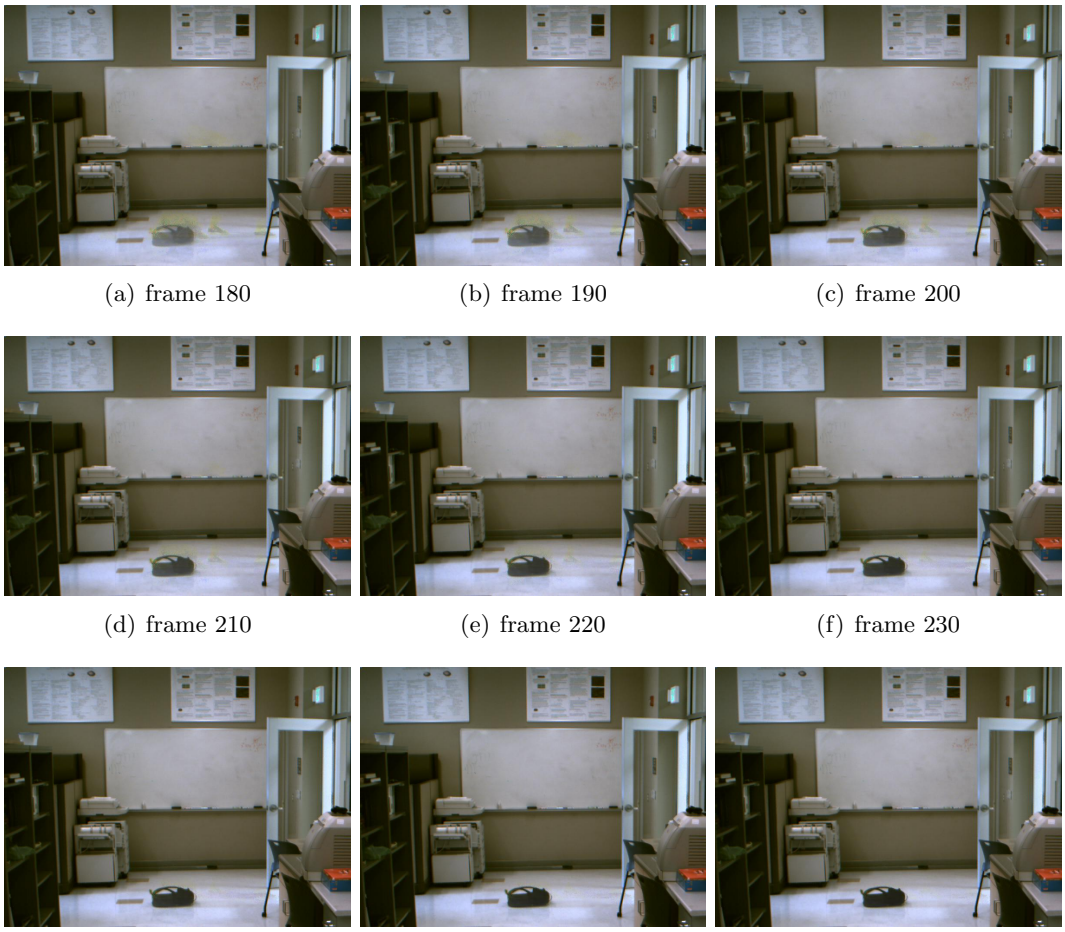


(a) frame 180 (b) frame 190 (c) frame 200

(d) frame 210 (e) frame 220 (f) frame 230

(g) frame 240 (h) frame 250 (i) frame 260

Fig. 4.23: Statistical dispersion background models of video sequence *backpack.avi*.

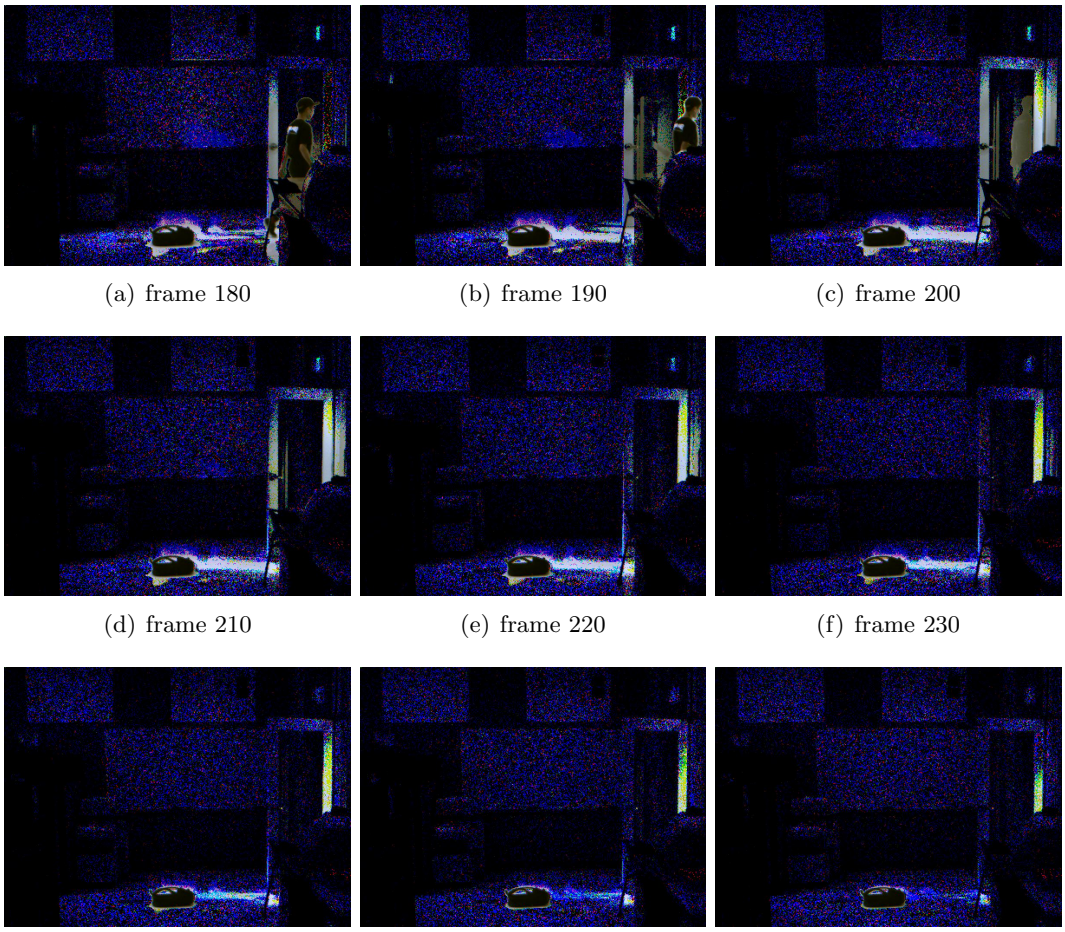


(a) frame 180 (b) frame 190 (c) frame 200

(d) frame 210 (e) frame 220 (f) frame 230

(g) frame 240 (h) frame 250 (i) frame 260

Fig. 4.24: Low rank and sparse background models of video sequence *backpack.avi*.



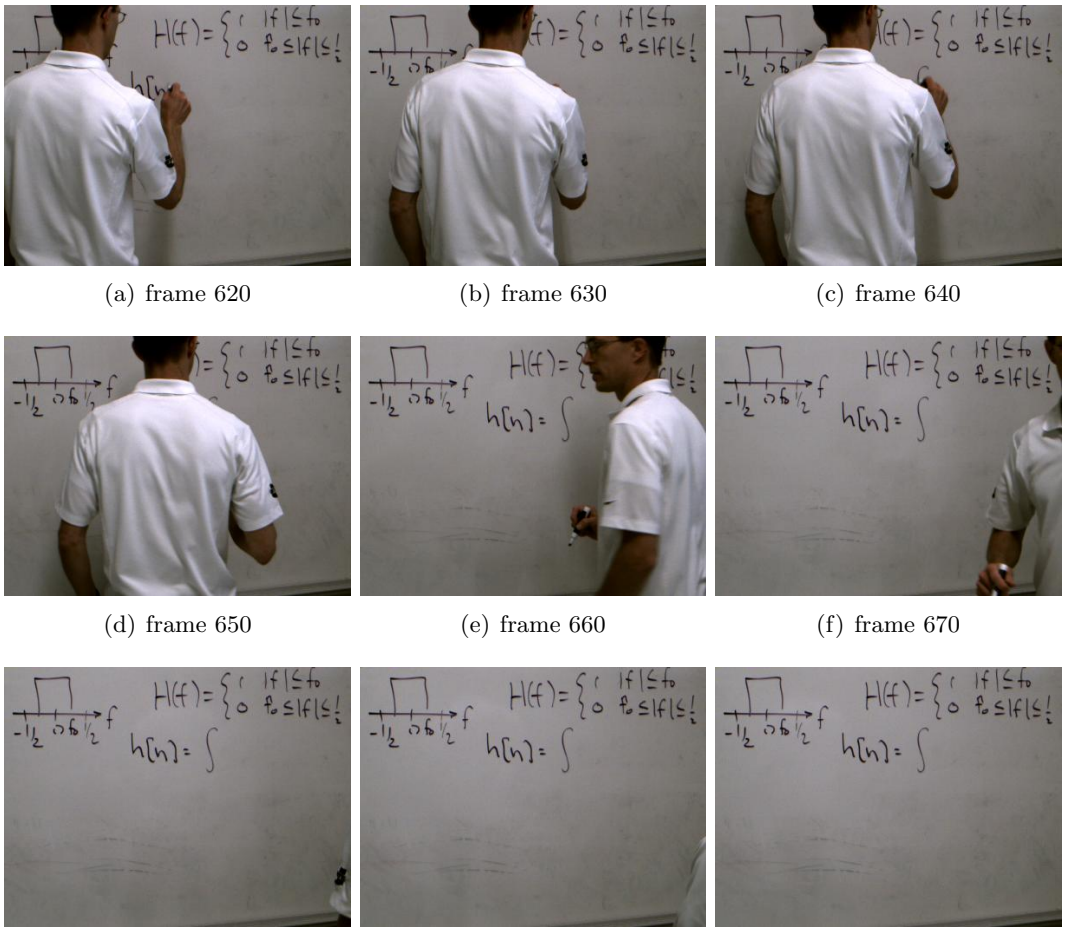
(a) frame 180 (b) frame 190 (c) frame 200

(d) frame 210 (e) frame 220 (f) frame 230

(g) frame 240 (h) frame 250 (i) frame 260

Fig. 4.25: Low rank and sparse foreground models of video sequence *backpack.avi*.

the adaptive linear prediction background models of the video sequence, Figure 4.28 is the statistical dispersion background models of the video sequence, Figure 4.29 is the low rank and sparse background models of the video sequence, and Figure 4.30 is the low rank and sparse foreground models of the video sequence.

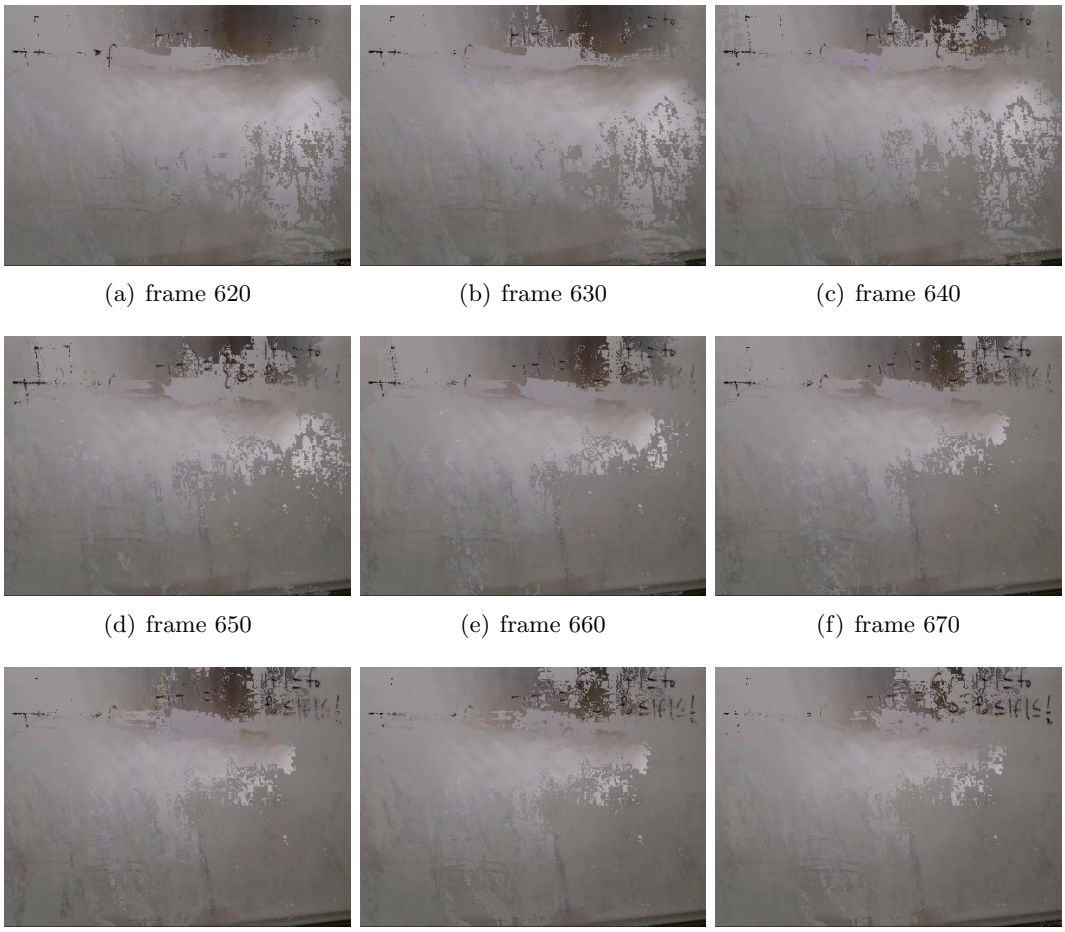


(a) frame 620 (b) frame 630 (c) frame 640

(d) frame 650 (e) frame 660 (f) frame 670

(g) frame 680 (h) frame 690 (i) frame 700

Fig. 4.26: Original frames of video sequence *lecture.avi*.

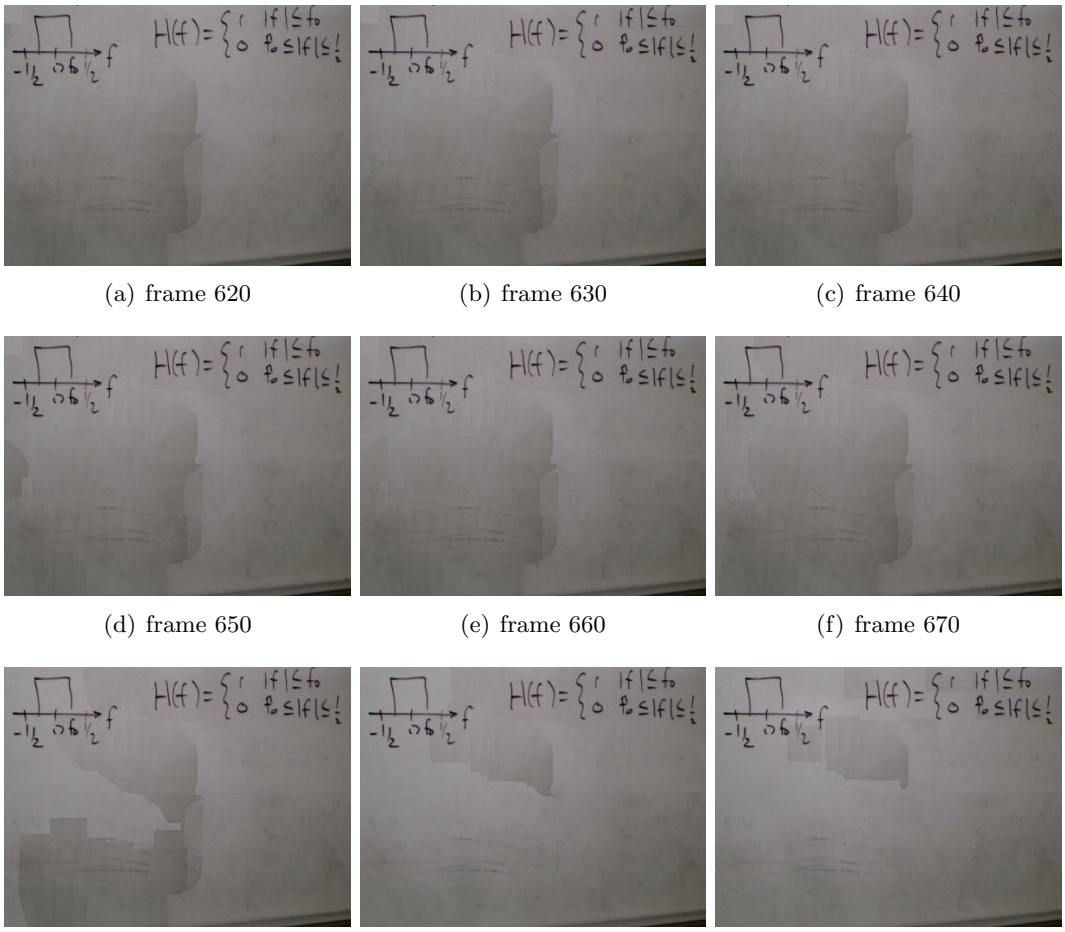


(a) frame 620 (b) frame 630 (c) frame 640

(d) frame 650 (e) frame 660 (f) frame 670

(g) frame 680 (h) frame 690 (i) frame 700

Fig. 4.27: Adaptive linear prediction background models of video sequence *lecture.avi*.

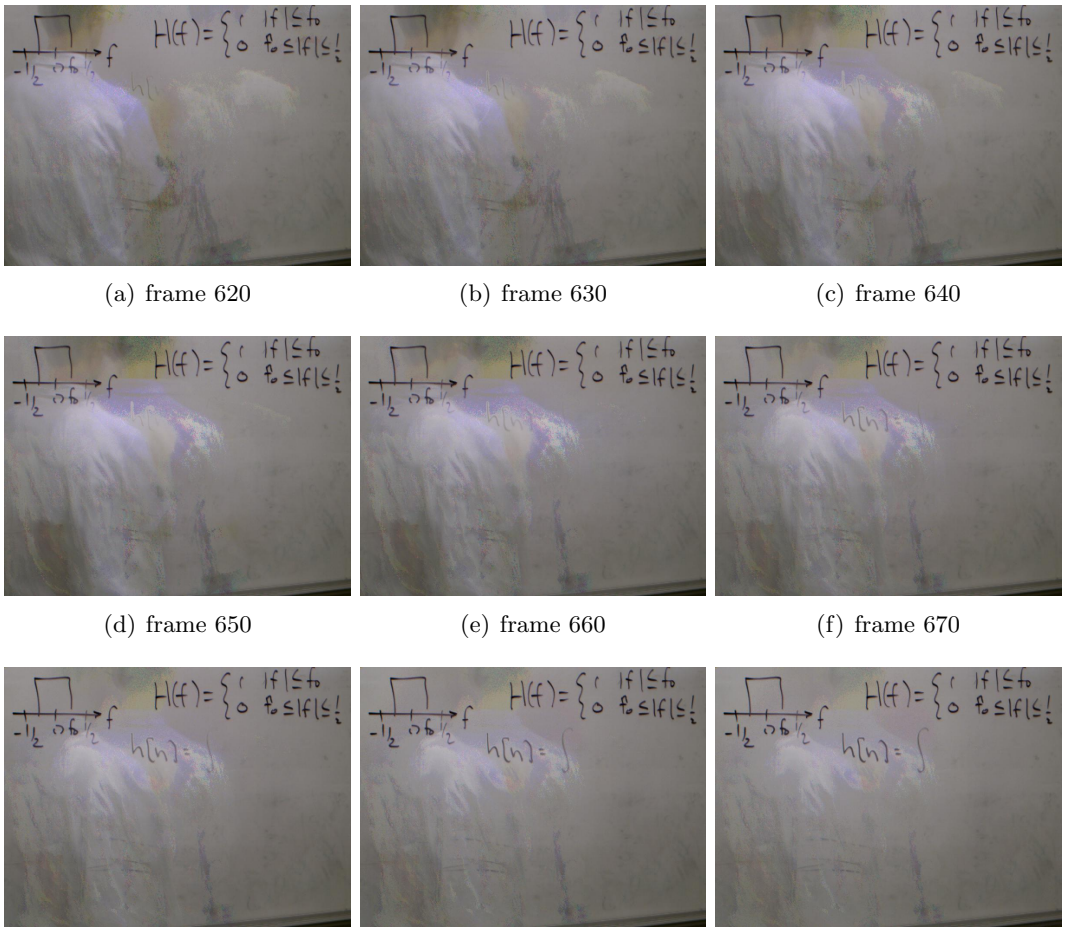


(a) frame 620 (b) frame 630 (c) frame 640

(d) frame 650 (e) frame 660 (f) frame 670

(g) frame 680 (h) frame 690 (i) frame 700

Fig. 4.28: Statistical dispersion background models of video sequence *lecture.avi*.

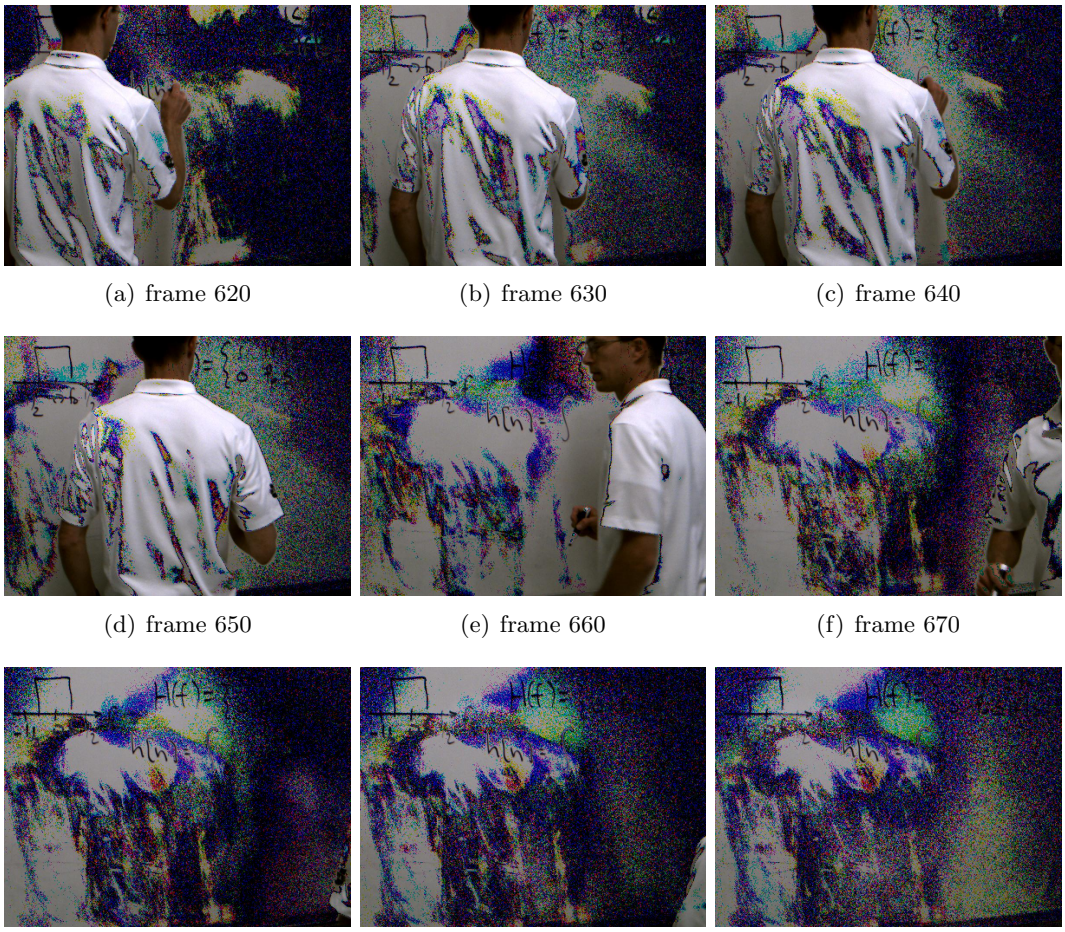


(a) frame 620 (b) frame 630 (c) frame 640

(d) frame 650 (e) frame 660 (f) frame 670

(g) frame 680 (h) frame 690 (i) frame 700

Fig. 4.29: Low rank and sparse background models of video sequence *lecture.avi*.



(a) frame 620 (b) frame 630 (c) frame 640

(d) frame 650 (e) frame 660 (f) frame 670

(g) frame 680 (h) frame 690 (i) frame 700

Fig. 4.30: Low rank and sparse foreground models of video sequence *lecture.avi*.

CHAPTER 5

Conclusion and Future Work

The goal of this research was to achieve adaptive background modeling with temporal feature updates in the form of developing "teacher removal" algorithms. Three algorithms were developed for this purpose. The first, adaptive linear prediction background modeling algorithm, was not effective for removing the teacher and keeping the writing on the whiteboard visible. Two factors added to the failure of this algorithm to model the adaptive background effectively and update temporal features. First, this algorithm relies on an initial estimation of a background without moving objects present, which was not present in the "lecture" video. Second, pixel-level processing utilized in the algorithm caused system instability, resulting in the video sequence never converging properly. However, this algorithm performed well with other forms of adaptive background modeling. It performed well when tested on the videos"vision traffic" and "atrium" because a static background was present at initialization.

The second, statistic dispersion background modeling algorithm, was the effective at removing the teacher, but parts of the writing were still obscured. Additionally, this algorithm assigns a stopped object to the background the fastest and removes a moving object from the background the fastest. However, a fair amount of ghosting was present, as well as the delay of the removal of black regions caused by initialization. If there is constant motion in a certain region, this region will be stuck at the initialization and never update. This algorithm worked well for all of the test videos besides "atrium" and "shopping mall" for the above mentioned reason – that is, constant motion in a specific region of the video sequences.

The third, low rank and sparse background modeling algorithm, was somewhat effective at removing the teacher but was most effective at allowing the writing on the whiteboard to be visible. The teacher was still highly visible, although transparent, because this algorithm

requires that the moving object be relatively smaller than the background. Despite this, the writing remained unobstructed and legible throughout the entirety of the video, and thus most closely achieved the initial goal. This algorithm also worked the best throughout all of the other test videos.

Because all three algorithms only require simple mathematical operations, the code can be implemented and optimized on a field-programmable gate array (FPGA) or a graphics processing unit (GPU) to be applied in real-time. The algorithm can also be improved by allowing hand-picked thresholds to be automated for adaption of the uncertain environments.

REFERENCES

[1] S.-C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Video Communications and Image Processing, SPIE Electronic Imaging*, vol. 200706. UCRL Conference San Jose, 2004.

[2] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proceedings of the 7th International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 255–261.

[3] J. Scott, M. A. Pusateri, and D. Cornish, "Kalman filter based video background estimation," in *Applied Imagery Pattern Recognition Workshop (AIPRW)*. IEEE, 2009, pp. 1–7.

[4] C. Stauffer and W. Crimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 1999, pp. 246–252.

[5] F. Zhang, L. Yang, and G. Zhang, "Adaptive fast gaussian background subtraction algorithm," in *Proceedings of the International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 2012, pp. 766–771.

[6] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 2. IEEE, 2004, pp. 28–31.

[7] P. Suo and Y. Wang, "An improved adaptive background modeling algorithm based on gaussian mixture model," in *Proceedings of the 9th International Conference on Signal Processing*, vol. 2. IEEE, 2008, pp. 1436–1439.

[8] X. Cui, J. Huang, S. Zhang, and D. N. Metaxas, "Background subtraction using group sparsity and low rank constraints," in *European Conference on Computer Vision (ECCV)*. ECCV, 2012.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition(CVPR)*. IEEE, 2005.

[10] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3. IEEE, 2011, pp. 500–513.

[11] E. J. Candes, X. Li, Y. Ma, and W. John, "Robust principal component analysis?" in *IEEE Transaction on Image Processing*. IEEE, 2009.

[12] J. Wright, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization," in *Neural Information Processing Systems*. NIPS, 2009.

[13] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, 2000.

[14] MATLAB, *version 8.3.0.532 (R2014a)*. Natick, Massachusetts: The MathWorks Inc., 2014.

[15] X. Li. (2016) Perception test images sequences @ONLINE. [Online]. Available: http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html