MOVING OBJECT DETECTION USING BIT PLANE SLICING

Nazgul Dastanova, Bachelor of Engineering

Submitted in fulfillment of the requirements
for the degree of Master of Science

School of Engineering
Department of Electrical and Electronic Engineering
Nazarbayev University

53 Kabanbay batyr Avenue,
Astana, Kazakhstan, 010000

December 9

Abstract

This thesis presents moving object detection algorithm using bit plane extraction of successive frames and comparing the respective bit planes by XOR operation. The proposed method works on 8-bit grayscale video frames obtained from a static camera. This algorithm is able to detect the motion of single and multiple objects in outside and inside environments.

Algorithm has been implemented in MATLAB by using several videos from VISOR database and was compared to existing conventional methods to show its effectiveness. Performance of an algorithm was evaluated based on ground truth metrics and results in terms of sensitivity, specificity, positive prediction and accuracy proved the validity of it. Results show that the proposed algorithm performs better in terms of mentioned metrics in comparison to other algorithms.

Acknowledgments

First of all, I would like to express my sincerest gratitude to my advisor, Professor Alex Pappachen James, for providing me support, guidance and sharing his knowledge with patience over the last several months. His positive learning environment, feedbacks and encouraging recommendations motivated me to work harder to improve the quality of the thesis.

Secondly, I would like to thank all my group mates and friends for making the process of study more interesting.

Lastly, I would like to thank my mother for her undoubted belief in me and support given throughout my life.

Contents

List of Figures

List of Tables

Chapter 1 – Introduction

1.1 Background

Motion detection can be described as the process of detecting pixel changes of the object with respect to the surrounding environment [1]. Although, motion detection can be implemented by various modalities, such as infrared, optical, and vibration-based, camera-based motion detection is prevalent since it allows computer vison-based techniques for processing [2]. Due to versatility, camera-based motion detection has been used extensively in traffic monitoring [3], video surveillance and people tracking [4]. Motion detection techniques are divided into three expansive classifications: Background Subtraction [5], Frame Differencing (Temporal Differencing) [6] and Optical Flow [7]. Nowadays, object detection by slicing the images into bit-planes is also attracting the interest. Digital image pixel values can be represented by bit sequences, where each bit plane illustrates the corresponding bit position [8]. The important visual data is stored in the higher order bits, while remaining bit planes give more discernible information [9]. Requiring less memory by discarding less significant bits, bit-plane extraction has found applications in biometrics (face [10], iris [11]and palm [12] recognition), image compression and biomedical image retrieval [13].

1.2 Thesis Motivation and Overview

The project focuses on novel moving object detection algorithm and the motivation lies in the intelligent video surveillance systems. Since billions of

devices are expected to be connected together over the Internet of Things (IoT) in the future, building the intelligent technologies is gaining interest among engineers. Proposed algorithm has already been implemented in hardware in Sultan Duisenbay's thesis. Developed small uncomplicated circuit along with pixel sensors can be applied for smart video systems. Algorithm has been done by comparing consecutive frames of the video. Extracted frames from video are converted into grayscale image to simplify the process. Then, two consecutive frames are taken to be compared by slicing them into bit-planes. Compared higher order (first four) bits are merged together and filtered to reduce the noise level. The process is repeated for the next two consecutive frames over the all video sequences.

1.3 Thesis outline

The rest of the thesis is organized as follows: Chapter 2 presents the review of the existing approaches in motion detection. Chapter 3 describes the approach of the proposed algorithm. Results from the MATLAB simulation and performance evaluation based on ground truth metrics are introduced in Chapter 4. Conclusions and open problems will be presented in Chapter 5.

Chapter 2 – Literature Review

Moving object detection is being interesting research area nowadays in computer vision field. The main goal of motion detection is to clearly identify the object and its path over the all frames of a video. Depending on whether background is static or dynamic, various methods and algorithms are used to analyze the videos. The steps of video examination procedure are: detecting the object of interest, tracking the detected object and observing the object behavior. In this chapter conventional algorithms, which will be compared with proposed technique, will be reviewed in detail.

As already mentioned, there are three expansive classifications of conventional motion detection approaches: Background Subtraction, Frame differencing (two- and three-frame differencing), and Optical Flow method. Among existing algorithms Background Subtraction method is gaining popularity due to its simplicity of implementation. The idea is to subtract the object from its background by applying the pre-processed background model. Step-by-step brief explanation of this method can be seen in algorithm pseudocode provided below.

Usually frame which does not contain any object is considered as background model. After obtaining difference frame between background model and current frame suitable threshold is applied to categorize pixels into foreground and background.

Algorithm 1 Background Subtraction algorithm pseudocode

```
repeat
    procedure ConvertingToGrayscale(all frames)
        for all frames do
            gray_frame ← convert_to_gray(frames)
        end for
    end procedure
    procedure ApplyingBackgroundModel(all frames)
        for all frames do
            backgroundmodel ← applying(background_model)
        end for
    end procedure
    procedure CalculatingTheAbsoluteDifference(frame_i, frame_background)
        for both frames do
            result_abs_diff ← get_abs_diff(frame_i_frame_background)
        end for
    end procedure
    procedure Filtering(result_abs_diff)
        result ← filtering(result_abs_diff)
        return result
    end procedure
    i ← i + 1
until frames_end
```

$$M(x,y) = \begin{cases} 1, foreground, & \text{if } |(i_t(x,y) - b(x,y)| > \mathrm{T} \\ 0, background & \text{if } |(i_t(x,y) - b(x,y)| < \mathrm{T} \end{cases} \tag{2.1}$$

where *M(x,y)* is the motion recognition mask, *i(x,y)* is the frame at time *t* and *b(x,y)* is the modelled background frame, *T* is the applied threshold [14]. Further object is detected by processing the image using suitable filter. This conventional method gives excellent results with a video from static camera. However, quality can be highly affected by dynamic background and outdoor environment. In order to improve the quality of detection, several methods have been developed. To illustrate, in [15] soft clustering and intensity histogram was

applied which successfully reduces the level of wrong failures. The drawback of the mentioned improvements is that they cannot manage the ghost effect and shadows. In another [16], pixel-wise background modelling by applying block-wise operator is proposed. Additional improvements into background difference method are introduced in [17], where color difference histogram along with Gaussian membership function suppresses the noise caused by background environment changes.

In Frame Differencing method, consecutive frames are compared to find the difference frame. As it is mentioned earlier, in this work two-frame differencing and three-frame differencing methods are analyzed.

Algorithm 2 Two-frame differencing algorithm pseudocode

```
1:  repeat
2:      procedure ConvertingToGrayscale(all frames)
3:          for all frames do
4:              gray_frame ← convert_to_gray(frames)
5:          end for
6:      end procedure
7:      procedure CalculatingTheAbsoluteDifference(frame_i, frame_{i+1})
8:          for both frames do
9:              result_abs_dif ← get_abs_dif(frame_i, frame_{i+1})
10:         end for
11:     end procedure
12:     procedure Filtering(result_abs_dif)
13:         result ← filtering(result_abs_dif)
14:         return result
15:     end procedure
16:     i ← i + 1
17: until frames_end
```

Two-frame differencing method is based on finding the motion from difference between adjacent frames [18]. In order to simplify the algorithm, the grayscale frames are processed. Then absolute difference of frames is deducted:

$$R_t(x, y) = |(Y_{(t+1)}(x, y) - Y_t(x, y)| \tag{2.2}$$

where, $Y_t(x,y)$ is the current frame at time t and $Y_{t+1}(x,y)$ is the frame at time $(t+1)$. Next step is to convert difference frame R_t to binary image by thresholding. If the intensity of a pixel on R_t is greater than given threshold, it is replaced with a white pixel and considered as detected motion or foreground, otherwise black pixel is put instead of corresponding pixel and considered as background.

$$B_t(x,y) = \begin{cases} 1, & \text{if } R_t(x, y) > \text{Th} \\ 0, & otherwise \end{cases} \tag{2.3}$$

The main issue of this method is low noise tolerance, so additional filtering is required such as application of median filter. Moreover, there are some additional disadvantages [19]:

- Important data can be lost due to thresholding method;
- Unsuitable for slow moving or stopping object;
- Ghost effect and aperture.

Ke Xiang et al. [20] proposed technique called Inter-frame Differencing image with Gaussian Mixture Models (IDGMM) where Two-frame Differencing along with the Gaussian Mixture Models (GMM) is used to solve the problem of sensitivity in segmenting methods. Results showed better overall performance and robustness compared to other conventional approaches. One of the most important problems in video surveillance is real-time motion detection. J.Cao

and L. Li [21] introduced intelligent traffic surveillance where Inter-frame differencing is applied to detect motion in dynamic background. In [22] inter-frame differencing combined with gamma correction was proposed for analyzing motion in low lightning condition by image correction. To overcome the problem of ghosting and to improve the overall performance of detection as an extension of this method Three-frame differencing approach was developed and it is described in Three-frame differencing algorithm pseudocode. [23] In this approach three frames, current, previous and next are taken to be analyzed. Original images also converted into grayscale images and after that absolute difference between current frame and previous frame as well between current frame and next frame is extracted.

$$R_{t,t-1}(x, y) = |(Y_t(x, y) - Y_{(t-1)}(x, y)| \tag{2.4}$$

$$R_{t,t+1}(x, y) = |(Y_{(t+1)}(x, y) - Y_t(x, y)| \tag{2.5}$$

where, $Y_t(x,y)$ is the current frame, $Y_{t-1}(x,y)$ is the frame at time $(t - 1)$ and $Y_{t+1}(x,y)$ is the frame at time $(t + 1)$. After finding both absolute difference frames suitable thresholds are applied to convert them into binary images and in the following formula 0 stands for background and 1 for foreground pixel values respectively.

$$B_{(t,t-1)}(x,y) = \begin{cases} 1, & \text{if } |(Y_t(x, y) - Y_{(t-1)}(x, y)| > \text{Th}_1 \\ 0, & otherwise \end{cases} \tag{2.6}$$

$$B_{(t+1,t)}(x,y) = \begin{cases} 1, & \text{if } |(Y_{(t+1)}(x,y) - Y_t(x,y)| > \mathrm{Th}_2 \\ 0, & otherwise \end{cases} \tag{2.7}$$

Then, integration of two binary images was done by using logical "AND" operator.

$$Integration(x,y) = \begin{cases} 1, & \text{if } (B_{(t,t-1)}(x,y) = B_{(t+1,t)}(x,y) = 1, \\ 0, & otherwise \end{cases} \tag{2.8}$$

Motion detection is then completed by processing the binary image by applying filters [24]. Three-frame differencing method joint with other approaches has gained wide range of use in moving object detection. In [25] object reconstruction algorithm by the application of linear operator for moving objects was proposed based on three-frame differencing method. By using this method one can improve the detection error rate value and velocity of detection, however, multiple object motion is the open problem for this method. Automatic thresholding in three-frame differencing algorithm can be introduced, also called as hybrid motion detection, which is able to select suitable threshold to simplify the task [26]. Although new approaches by using three-frame differencing method have been developed, there are still unsolved issues such as occlusion, background noise, illumination problems and etc [24].

One of the computationally complex approaches in moving object detection is the Optical Flow method. Based on [26], pseudocode of this method is shown below. In this method two successive frames are processed and each pixel is considered as vector and called optical flow. It gives displacement of each

Algorithm 3 Three-frame differencing algorithm pseudocode

```
repeat
    procedure ConvertingToGrayscale(all frames)
        for all frames do
            gray_frame ← convert_to_gray(frames)
        end for
    end procedure
    procedure CalculatingTheAbsoluteDifference(frame_{i-1}, frame_i)
        for both frames do
            result_abs_dif1 ← get_abs_dif(frame_{i-1}_frame_i)
        end for
    end procedure
    procedure CalculatingTheAbsoluteDifference(frame_i, frame_{i+1})
        for both frames do
            result_abs_dif2 ← get_abs_dif(frame_i_frame_{i+1})
        end for
    end procedure
    procedure AND(abs_dif1, abs_dif2)
        for both frames do
            result_combined ← AND(abs_dif1, abs_dif2)
        end for
    end procedure
    procedure Filtering(result_combined)
        result ← filtering(result_combined)
        return result
    end procedure
    i ← i + 1
until frames_end
```

pixel compared to previous pixel over time. Due to the complexity of calculations one can transfer 3D pictures to 2D pictures and use brightness constancy assumption:

$$f(x, y, t) = f(x + dx, y + dy, t + dt), \tag{2.9}$$

Employing Taylor series for right part of (2.9) equation and adding some changes to obtained equation gives equation of optical flow interpretation:

$$f_x u + f_y v + f_y = 0, \tag{2.10}$$

Algorithm 4 Optical Flow Algorithm pseudocode for motion detection

```
repeat
    procedure ConvertingToGrayscale(all frames)
        for all frames do
            gray_frame ← convert_to_gray(frames)
        end for
    end procedure
    procedure OpticalFlowEstimation(frame_i, frame_{i+1})
        for both frames do
            result_optical_flow ← get_optical_flow_estimation(frame_i_frame_{i+1})
        end for
    end procedure
    procedure Filtering(result_optical_flow)
        result ← filtering(result_optical_flow)
        return result
    end procedure
    i ← i + 1
until frames_end
```

It can be presented also in a vector form:

$$\nabla f \vec{v} = -f_t, \tag{2.11}$$

where ∇f is brightness intensity spatial gradient, $\vec{v}$ is the speed vector of a pixel and f_t is the time derivative of brightness intensity. (2.10) equation is mostly used in optical flow estimation and known as the gradient constraint [27]. In this case we cannot calculate two variables (u and v) and it is referred to aperture issue. Widely used techniques for calculating optical flow are: Lucas-Kanade and Horn-Schunck [27].

Recently bit plane slicing of an image has also been used for moving object detection. Through the use of binary digits one can represent pixels of a digital image. Alternately, each bit plane illustrates the corresponding bit position of the binary digit. The number of bit planes is the same with the number of bits of the binary digits. For instance, in 8-bit grayscale image, there are 8 bit planes

since original image uses 8 bits per pixel. Numeric value of bits has positive and negative meanings, "1" and "0" respectively. The uttermost value of a bit (bit plane 7) which can have significant effect on the pixel is known as most significant bit (MSB), whereas, in contrast, lowest value (bit plane 0) has minor impact on the pixel and termed as least significant bit (LSB). Visibly important data is in the first higher order bits (higher four), remaining bit planes give more detailed information. Bit plane slicing is a process of extraction of these bits [28]. In [29] algorithm for motion detection was introduced and the main goal was achieved by using bit plane slicing,hysteresis thresholding, memorizing the motion and BLOB analyses. This algorithm succeeds in overcoming the issue of edge loss and it is applicable for real-time video surveillance systems. The open problem for this approach is to reduce the storage employed by motion history.

Chapter 3 – Methodology

3.1 Working environment and algorithm overview

In this thesis proposed algorithm simulation was implemented in MATLAB software environment for extracting video frames and further processing. In order to show the effectiveness of proposed method three different videos from VISOR (Video surveillance online repository) database was chosen.VISOR is a set of freely available videos for research purposes on pattern recognition and multimedia retrieval.VISOR also includes the metadata annotation, both manually obtained by the use of ground truth and automatically obtained by video systems. [30] Experiments were conducted for outside and inside, also single and multiple motion cases. The algorithm pseudocode is represented below.

As it can be seen in proposed algorithm pseudocode, the input images are acquired from videos and pre-processed by converting them into grayscale images.The reason of using grayscale images is that it gives an opportunity to make the process simpler as we take into account only one channel of pixels. Detection of moving object is achieved by breaking down the current and previous frames into constitutive bit planes. As a result, bit planes are acquired for each of the frames. Respective bit planes are compared by XOR operation for each of the frames (i.e. most significant bit (MSB) bit plane of current frame is compared to MSB bit plane of the previous frame). Important features are encoded into higher order bit planes, and less significant bit planes carry discernible information. The resultant bit planes, formed by XOR comparison, are then combined to form a grayscale image of the moving object. Combination of bit planes is

Algorithm 5 Proposed Algorithm pseudocode for motion detection

```
repeat
    procedure BitExtraction(frame_i, frame_{i+1})
        for both frames do
            gray_frame ← convert_to_gray(frame)
            for all bit planes do
                b_p ← get_bit_planes(gray_frame)
            end for
        end for
    end procedure
    procedure XOR(frame_i, frame_{i+1})
        for both frames do
            result_b_p ← XOR(b_p_i, b_p_{i+1})
        end for
        for high order bit planes do
            gray_result ← combine(result_b_p)
        end for
    end procedure
    procedure Filtering(gray_result)
        result ← filtering(gray_result)
        return result
    end procedure
    i ← i + 1
until frames_end
```

realized by applying weights, which correspond to bit plane order. By applying morphological operator and thresholding process, combined grayscale image is converted into binary image with the detected moving object.

3.2 Bit plane extraction

The input color images are acquired from downloaded videos, pre-processed and converted into 8-bit grayscale images.The number of bit planes is equal to the number of digital bits of the image. Hence, in 8-bit grayscale image there are 8 bit planes. Grayscale pixel values of processed image is further sliced into

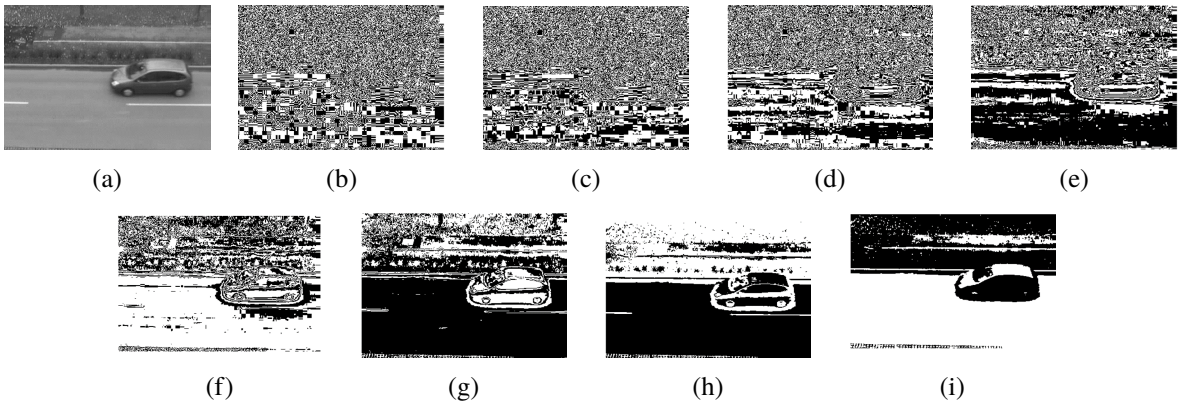


Figure 3.1: Original grayscale image and bit planes of the 3rd frame of Video 1. (a) Original image, (b) LSB, (c) 1st bit, (d) 2nd bit, (e) 3rd bit, (f) 4th bit, (g) 5th bit, (h) 6th bit, (i) MSB

eight constituting bits by:

$$\lfloor \frac{Y}{2^k} \rfloor mod 2 = a_k, \tag{3.1}$$

where Y is grayscale pixel value, k is the bit number, mod is a modulo operation, and a_k is the bit value of the corresponding bit number. $\lfloor \cdot \rfloor$ operation is represented by:

$$\lfloor x \rfloor = m \iff m \leq x < m + 1, \tag{3.2}$$

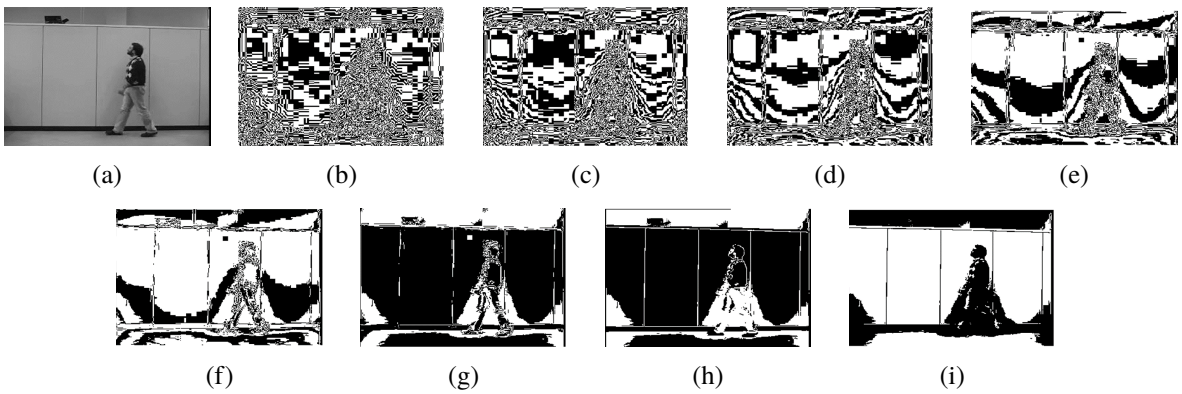


Figure 3.2: Original grayscale image and bit planes of the 149th frame of Video 2. (a) Original image, (b) LSB, (c) 1st bit, (d) 2nd bit, (e) 3rd bit, (f) 4th bit, (g) 5th bit, (h) 6th bit, (i) MSB

Each of the acquired bits is then stored in separate matrices for each bit

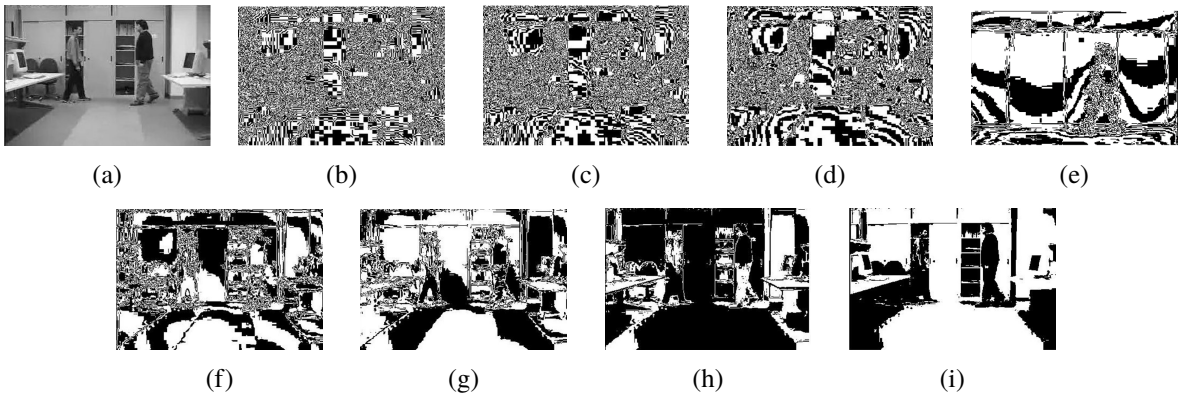


Figure 3.3: Original grayscale image and bit planes of the 162nd frame of Video 3. (a) Original image, (b) LSB, (c) 1st bit, (d) 2nd bit, (e) 3rd bit, (f) 4th bit, (g) 5th bit, (h) 6th bit, (i) MSB

plane. Since lower order bit planes do not convey important information, only higher order bit planes are stored for further processing. This results in less memory utilization. Bit planes of a target grayscale image of different video sequences are represented in Fig. 3.1, Fig. 3.2, Fig. 3.3.

3.3 Comparison of bit planes

The respective bit planes of two consecutive frames are compared via XOR process respectively (MSB plane of past frame and MSB plane of current frame). The same process occurs with remaining bit planes. Comparison by using XOR process can be thought of as a simpler version of absolute difference process used in the previous methods, where bits are considered instead of a three digit number (0-255). XOR process is given by:

$$a_k \oplus b_k = c_k, \tag{3.3}$$

where a_k, b_k, c_k are k-th bit values of 1st frame pixel, 2nd frame pixel, and resultant bit value respectively. The results of bit plane comparison are illustrated

in Fig. 3.4, Fig. 3.5, Fig. 3.6.

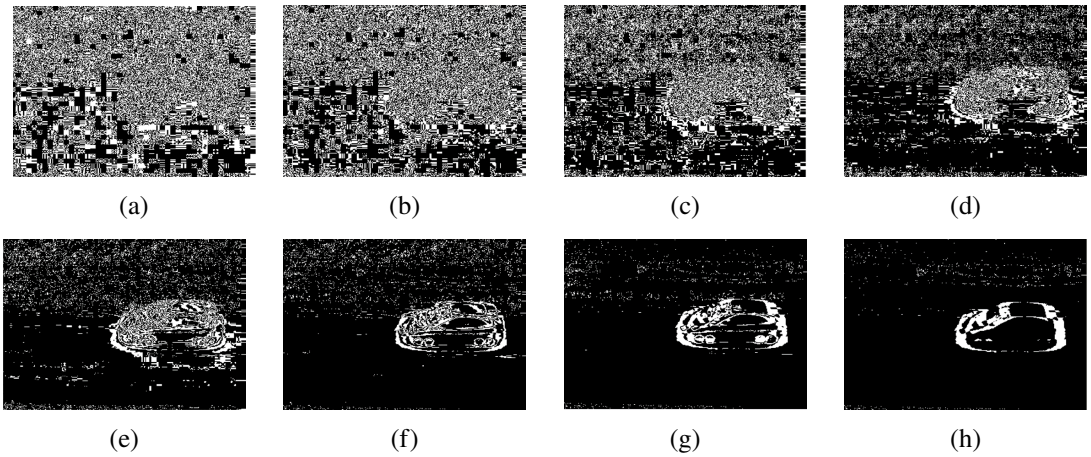


Figure 3.4: Comparison of 3rd and 4th frames' bit planes via XOR operation of Video 1. (a) LSB, (b) 1st bit, (c) 2nd bit, (d) 3rd bit, (e) 4th bit, (f) 5th bit, (g) 6th bit, (h) MSB

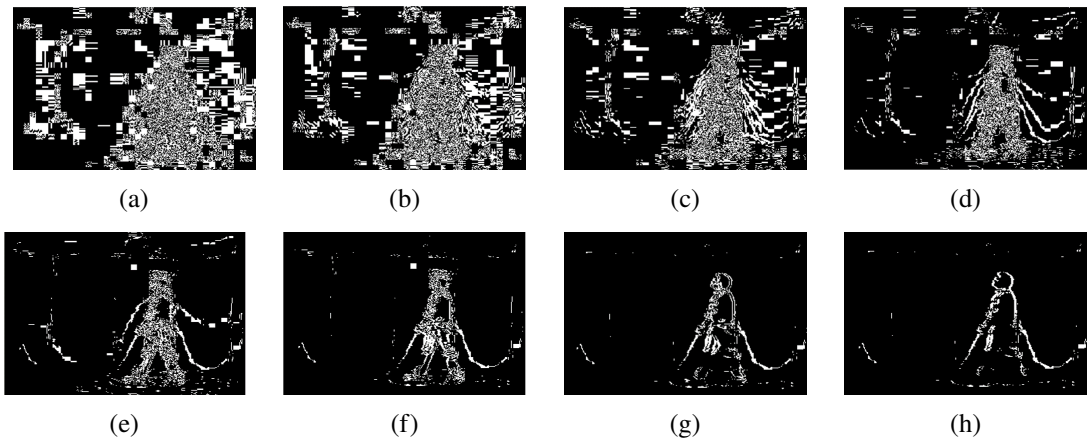


Figure 3.5: Comparison of 149th and 150th frames' bit planes via XOR operation of Video 2. (a) LSB, (b) 1st bit, (c) 2nd bit, (d) 3rd bit, (e) 4th bit, (f) 5th bit, (g) 6th bit, (h) MSB

3.4 Processing and detection

After comparison of bit planes, higher bit planes are merged together again to obtain a grayscale image by:

$$\sum_{k=4}^{7} 2^k * c_k = Y, \tag{3.4}$$

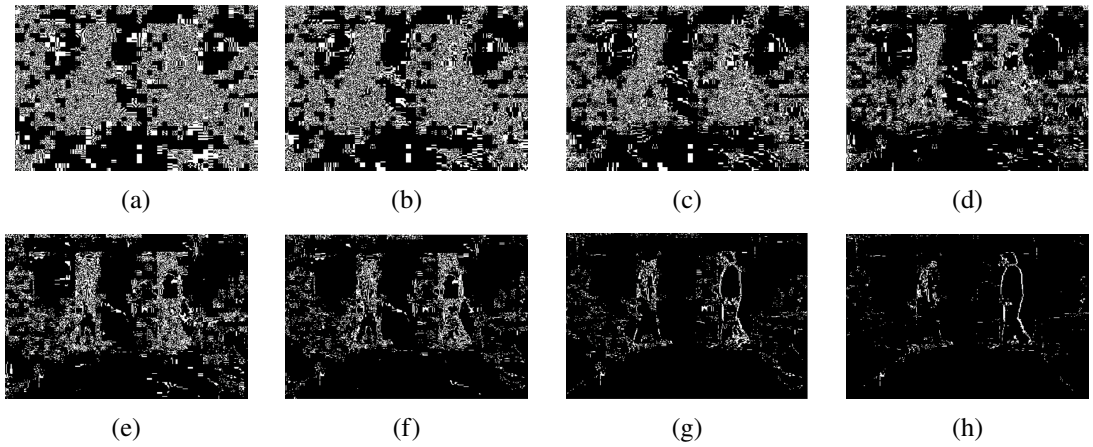


Figure 3.6: Comparison of 162nd and 163th frames' bit planes via XOR operation of Video 3. (a) LSB, (b) 1st bit, (c) 2nd bit, (d) 3rd bit, (e) 4th bit, (f) 5th bit, (g) 6th bit, (h) MSB

where k is the bit number, c_k is the bit value, and Y is the resultant grayscale pixel value.

Grayscale image is formed from weighted combination of four higher order bit planes, which make contribute significantly to image formation, according to formula above. The obtained grayscale image is further filtered to detect the object. Median filter is applied to combined grayscale image to remove the noise by preserving the object edges. The main principle of this filter is finding median value of image pixels by analyzing neighborhood pixels. Figure 3.7 clearly illustrates the difference between a combined image and filtered one.

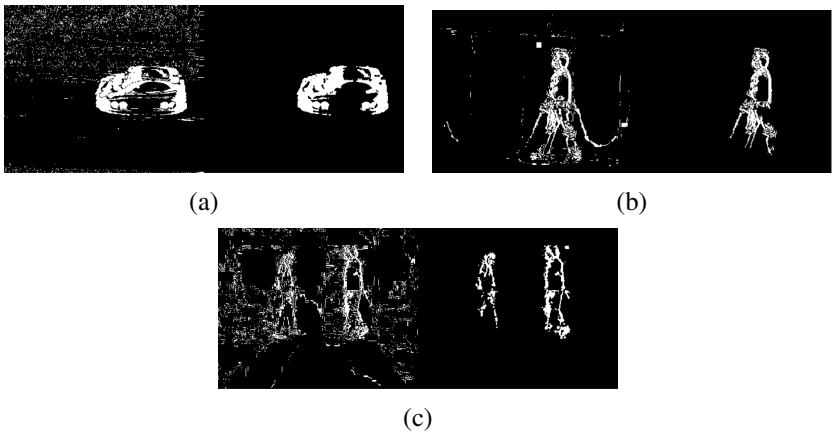


Figure 3.7: Detection of the moving objects in Videos. Original image acquired from combination of XORed bit planes and Processed image.

Chapter 4 – Results and Discussion

In this section comparison results of proposed method with other four conventional methods (Two-frame differencing, Three-frame differencing, Background Subtraction, Optical flow) is represented. MATLAB codes of all algorithms for simulation can be seen in Appendices. Experiments are conducted on videos recorded in different conditions such as outdoor and indoor environments, also videos containing one and multiple objects are taken into account to show the effectiveness of proposed method. Different approaches have been examined on different video sequences:

1. Video 1: Moving cars on a road captured by a traffic video surveillance system. Video quality is 368 pixels x 288 pixels.
2. Video 2: A person enters to the room, rises his hand and goes forward. Video quality is 320 pixels x 240 pixels.
3. Video 3: Two men come from opposite directions, stop at the center of the room. Video quality is 384 pixels x 288 pixels.

Several video frames are taken to be tested and simulation results are shown in Fig. 4.2, Fig. 4.3, Fig. 4.4. Figures contain original images and detected objects by using five different methods including proposed algorithm. Results are visualized on three frames from each video. Step-by-step results will be described below in detail.

4.1 Performance evaluation metrics

Quantitative examinations in terms of accuracy, sensitivity, specificity, tracker detection rate and etc. are obtained by the use of ground truth measurements. Frame based ground truth method is used to compare each frame separately with ground truth frame in terms of intensity, position and quantity of objects, without considering the similarity of the objects over the all video sequences. In this work 200 frames from each video are taken to be tested. Ground truth measurements can be briefly explained in Fig. 4.1:

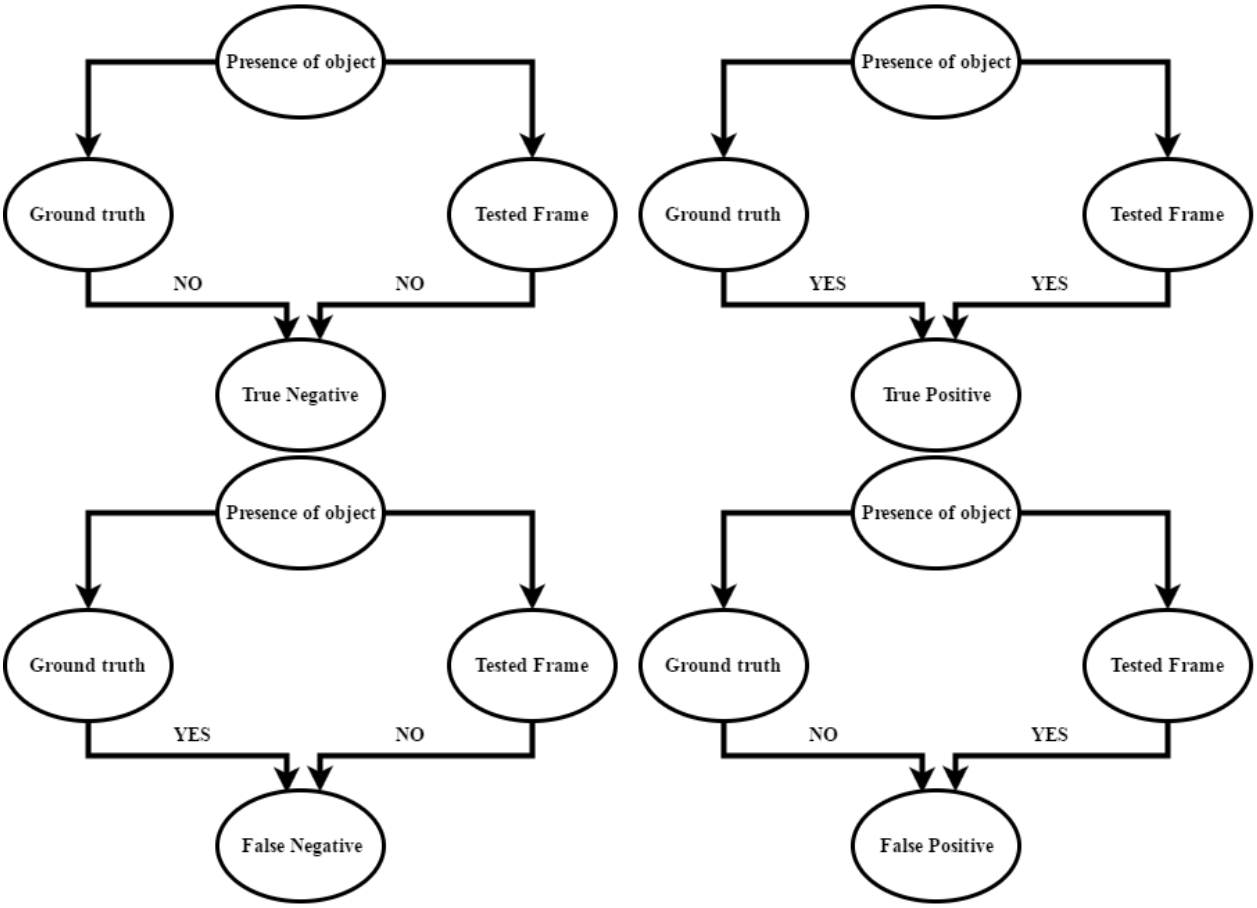


Figure 4.1: The logic of performance evaluation based on ground truth metrics

Let TG be the overall number of video sequences where ground truth

frame contains the object and TF is the overall number of tested video frames. Based on these values further measurements can be calculated and it should be noted that TP stands for True Positive, TN for True negative, FN for False negative and FP for False positive, respectively:

$$TrackerDetectionRate(TDR) = \frac{TP}{TG} \tag{4.1}$$

$$FalseAlarmRate(FAR) = \frac{FP}{TP + FP} \tag{4.2}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.3}$$

$$Specificity = \frac{TN}{FP + TN} \tag{4.4}$$

$$Accuracy = \frac{TP + TN}{TF} \tag{4.5}$$

$$PositivePrediction(PP) = \frac{TP}{TP + FP} \tag{4.6}$$

$$NegativePrediction(NP) = \frac{TN}{FN + TN} \tag{4.7}$$

$$FalseNegativeRate(FNR) = \frac{FN}{FN + TP} \tag{4.8}$$

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} \tag{4.9}$$

4.2 Performance evaluation

In order to take into account the noise that can be generated by outdoor environment, all the methods were applied to the video from traffic surveillance camera (Video 1). Results can be seen Fig. 4.2 .

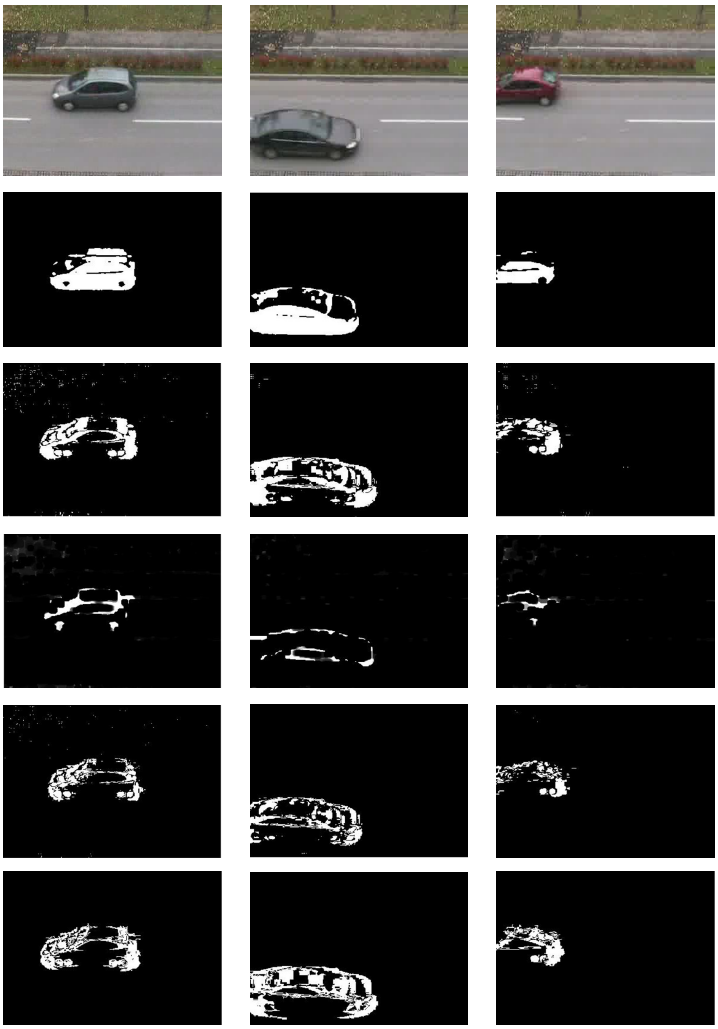


Figure 4.2: The comparison between proposed algorithm and existing methods. From top to bottom: Original images, Two-frame differencing, Three-frame differencing, Background Subtraction, Optical Flow, Proposed Method.(a) 10th frame,(b) 122nd frame, (c)158th frame

Table 4.1 represents the results received from tested outdoor environment video. Although, in outdoor environment there are several interferences, such as moving tree leaves, flying birds, windy weather, which dramatically affect the quality of tracking, all compared methods, including the proposed algorithm, give satisfactory results. More rigorously, all methods displayed excellent results in tracker detection rate, sensitivity and false negative rate metrics. Considering the false alarm rate, specificity, accuracy and positive rate metrics, the proposed method significantly outperforms background subtraction and optical flow meth-

Table 4.1: Comparison of quantitative results between proposed method and existing methods for outdoor motion detection case

Measurements	Two-frame	Three-frame	BS	OF	Proposed Algorithm
TDR	0.985	0.978	1	1	0.985
FAR	0.028	0.014	0.297	0.301	0.014
Sensitivity	1	0.992	1	1	1
Specificity	0.935	0.964	0.033	0	0.968
Accuracy	0.975	0.975	0.705	0.695	0.985
PP	0.971	0.969	0.702	0.698	0.985
NP	1	0.983	1	0	1
FNR	0	0.007	0	0	0
FPR	0.064	0.033	0.967	1	0.032

ods, while two-frame and three-frame differencing methods show comparable results. In addition to this, the proposed method displays slightly better performance compared to all other methods in accuracy and positive prediction metrics.

Second test video included a person walking into the room, stopping and raising his arm. This video was chosen in order to observe the performance of all algorithms in the environment with negligible interferences. Results can be seen in Fig. 4.3.

Table 4.2 provides results on the algorithms' performance for the second video. Focusing on tracker detection rate, sensitivity, false negative rate and accuracy metrics, the proposed method along with background subtraction and optical flow displayed slightly better results than the inter-frame differencing methods. All the observed methods performed well in terms of positive prediction and false alarm rate. The proposed algorithm outperformed other methods in negative prediction with 100% efficiency, while the second best negative prediction result was obtained from optical flow method (66.7%).

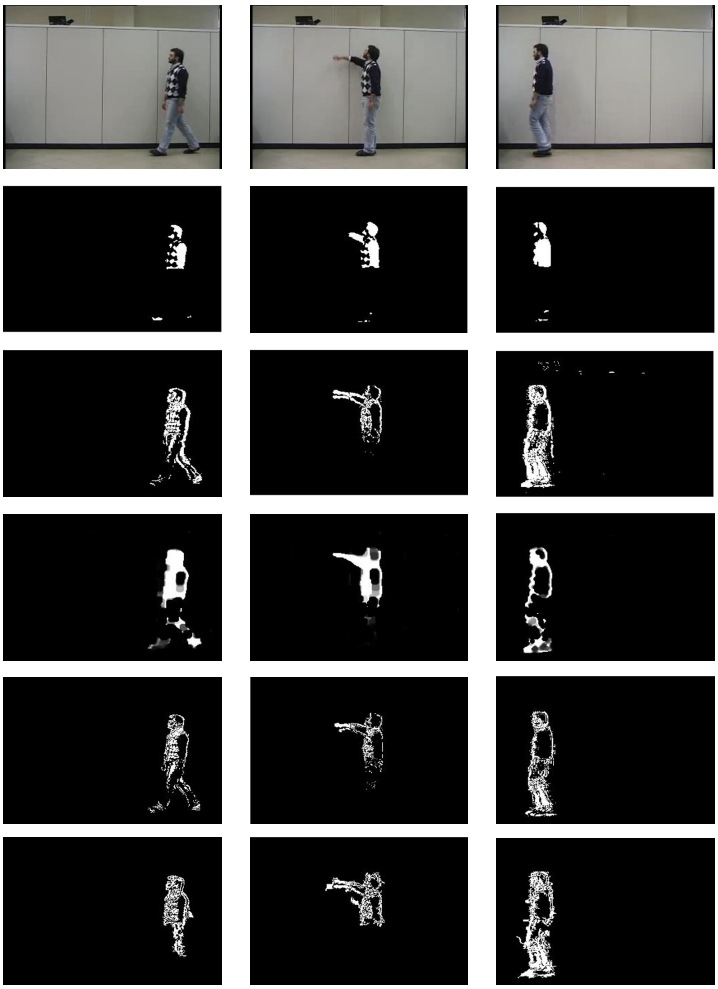


Figure 4.3: The comparison between proposed algorithm and existing methods.From top to down: Original images, Two-frame differencing, Three-frame differencing, Background Subtraction, Optical Flow, Proposed Method.(a) 134th frame,(b) 162nd frame, (c)238th frame

Comparative performance of moving object detection algorithms was also checked on the video containing multiple moving objects. The last video (Video 3) includes two men walking towards each other with the foreground furniture and stationery. Results are depicted in Fig. 4.4.

Evaluation of algorithms' performance using ground truth metrics is given in Table 4.3. Results show that all algorithms, excluding background subtraction, perform excellently in tracker detection rate, sensitivity and false negative rate. However, the proposed method clearly outperformed other methods in

Table 4.2: Comparison of quantitative results between proposed method and existing methods for indoor single motion detection case

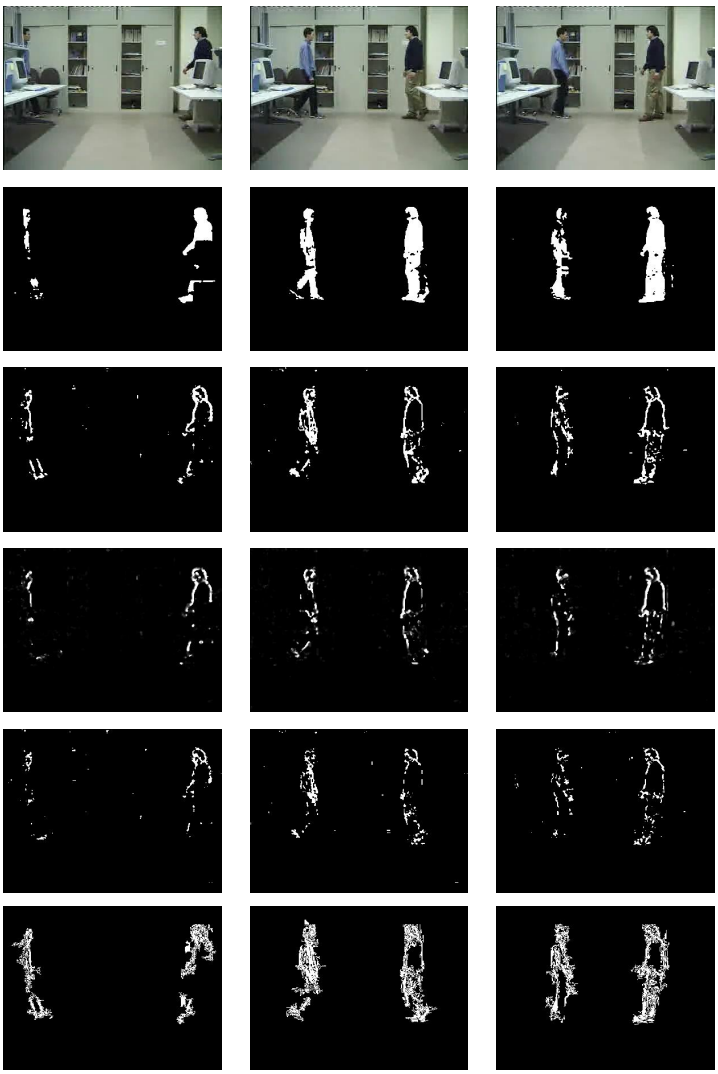
Measurements	Two-frame	Three-frame	BS	OF	Proposed Algorithm
TDR	0.876	0.788	1	0.976	0.994
FAR	0	0	0	0	0
Sensitivity	0.882	0.797	1	0.982	0.994
Specificity	1	1	0	1	1
Accuracy	0.88	0.795	0.965	0.977	0.994
PP	1	1	0.966	1	1
NP	0.231	0.15	0	0.667	1
FNR	0.188	0.202	0	0.018	0
FPR	0	0	1	0	0

Table 4.3: Comparison of quantitative results between proposed method and existing methods for indoor multiple motion detection case

Measurements	Two-frame	Three-frame	BS	OF	Proposed Algorithm
TDR	1	1	0.553	1	0.985
FAR	0.527	0.525	0.188	0.523	0.014
Sensitivity	1	1	0.553	1	1
Specificity	0	0.019	0.886	0.019	0.967
Accuracy	0.472	0.479	0.729	0.482	0.985
PP	0.472	0.474	0.813	0.477	0.985
NP	0	1	0.689	1	1
FNR	0	0	0.446	0	0
FPR	1	0.98	0.114	0.98	0.03

specificity, accuracy, positive prediction and false positive rate measurements.

The results show that the proposed method outperformed the mentioned algorithms on average, with some metrics sharing the comparable value. In addition to this, it is seen algorithm performance is heavily impacted by the video itself (outdoor or indoor, single object or multiple objects).



(a) (b) (c)

Figure 4.4: The comparison between proposed algorithm and existing methods.From top to down: Original images, Two-frame differencing, Three-frame differencing, Background Subtraction, Optical Flow, Proposed Method.(a) 122nd frame,(b) 147th frame, (c)152nd frame

Chapter 5 – Conclusion and Future work

The goal of this thesis was to implement novel moving object detection via bit plane slicing. Proposed algorithm is implemented in MATLAB simulation environment. Comparison examinations with other conventional methods are conducted on Visor database videos. Performance of the algorithm is evaluated based on ground truth metrics.

Experimental results prove that proposed algorithm demonstrates slightly better performance on average compared to other conventional methods. However, in most cases the proposed method showed comparable results in terms of metrics.

The main advantage of the proposed method lies in achieving the less memory utilization via bit plane slicing. In interframe differencing and background subtraction methods 8-bit grayscale pixel values are used to perform comparison. The proposed method, however, utilizes the fact that important information is concentrated in higher order bits and uses only four higher bits.

Moreover, this algorithm can be implemented in hardware by building uncomplicated small circuit and along with pixel sensors can be applied to smart video surveillance systems. It should be underlined that hardware implementation of proposed algorithm is more thoroughly discussed in Sultan Duisenbay's work.

In the future work, as an extension of this proposed method one can handle the problem of dynamic background and the issue of application to real-time video surveillance systems. As well as that developing the complete prototype of the video surveillance system based on this algorithm is highly recommended.

Bibliography

[1] M. Zhu, S. Sun, S. Han, and H. Shen, "Comparison of moving object detection algorithms," in *World Automation Congress 2012*, June 2012, pp. 35–38.

[2] J. Dey and N. Praveen, "Moving object detection using genetic algorithm for traffic surveillance," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, March 2016, pp. 2289–2293.

[3] N. Raviprakash, M. Suresh, A. Rathis, D. Devarla, A. Yadav, and G. S. Nagaraja, "Moving object detection for content based video retrieval," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, April 2016, pp. 0322–0326.

[4] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1820–1833, Sept 2011.

[5] O. Barnich and M. V. Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, June 2011.

[6] Y. Lin, M. Fang, and D. Shihong, "An object reconstruction algorithm for moving vehicle detection based on three-frame differencing," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE*

12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), Aug 2015, pp. 1864–1868.

[7] X. Han, Y. Gao, Z. Lu, Z. Zhang, and D. Niu, "Research on moving object detection algorithm based on improved three frame difference method and optical flow," in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, Sept 2015, pp. 580–584.

[8] F. Francis-Lothai and D. B. L. Bong, "An analysis of the effects of bit plane extraction in fingerprint recognition," in *2014 IEEE Conference on Systems, Process and Control (ICSPC 2014)*, Dec 2014, pp. 132–136.

[9] K. C. Ting, D. B. L. Bong, and Y. C. Wang, "Performance analysis of single and combined bit-planes feature extraction for recognition in face expression database," in *2008 International Conference on Computer and Communication Engineering*, May 2008, pp. 792–795.

[10] T. Z. Lee and D. B. L. Bong, "Face and palmprint multimodal biometric system based on bit-plane decomposition approach," in *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, May 2016, pp. 1–2.

[11] B. Bonney, R. Ives, D. Etter, and Y. Du, "Iris pattern extraction using bit planes and standard deviations," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, vol. 1, Nov 2004, pp. 582–586 Vol.1.

[12] R. Sammouda, H. B. Mathkour, and A. Touir, “A modified bit-plane based method for lung region extraction from 3d chest ct images,” in *2013 Second International Japan-Egypt Conference on Electronics, Communications and Computers (JEC-ECC)*, Dec 2013, pp. 34–39.

[13] F. Francis-Lothai and D. B. L. Bong, “An analysis of the effects of bit plane extraction in fingerprint recognition,” in *2014 IEEE Conference on Systems, Process and Control (ICSPC 2014)*, Dec 2014, pp. 132–136.

[14] T. Kiran and G.-R. Kwon, “An advanced segmentation using bit-plane slicing technique in extraction of lungs region,” in *2011 Second Asian Himalayas International Conference on Internet (AH-ICI)*, Nov 2011, pp. 1–5.

[15] J. Lee, S. Lim, J. G. Kim, B. Kim, and D. Lee, “Moving object detection using background subtraction and motion depth detection in depth image sequences,” in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, June 2014, pp. 1–2.

[16] W. Liu, H. Yu, H. Yuan, H. Zhao, and X. Xu, “Effective background modelling and subtraction approach for moving object detection,” *IET Computer Vision*, vol. 9, no. 1, pp. 13–24, 2015.

[17] D. K. Panda and S. Meher, “Detection of moving objects using fuzzy color difference histogram based background subtraction,” *IEEE Signal Processing Letters*, vol. 23, no. 1, pp. 45–49, Jan 2016.

[18] L. Christodoulou, T. Kasparis, and O. Marques, “Advanced statistical and adaptive threshold techniques for moving object detection and segmenta-

tion," in *2011 17th International Conference on Digital Signal Processing (DSP)*, July 2011, pp. 1–6.

[19] I. Kartika and S. S. Mohamed, "Frame differencing with post-processing techniques for moving object detection in outdoor environment," in *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, March 2011, pp. 172–176.

[20] K. Xiang, X. Wang, S. Cao, and X. Fu, "A new approach for real-time segmenting moving objects under cluttered background," in *2012 IEEE Symposium on Electrical Electronics Engineering (EEESYM)*, June 2012, pp. 224–227.

[21] J. Cao and L. Li, "Vehicle objects detection of video images based on gray-scale characteristics," in *2009 First International Workshop on Education Technology and Computer Science*, vol. 2, March 2009, pp. 936–940.

[22] Y. Miura and Y. Fujii, "The examination of the image correction of the moving-object detection for low illumination video image," in *2015 IEEE International Conference on Consumer Electronics - Taiwan*, June 2015, pp. 35–36.

[23] H. Zhang and K. Wu, "A vehicle detection algorithm based on three-frame differencing and background subtraction," in *2012 Fifth International Symposium on Computational Intelligence and Design*, vol. 1, Oct 2012, pp. 148–151.

[24] S. Murali and R. Girisha, "Segmentation of motion objects from surveillance video sequences using temporal differencing combined with multiple

correlation," in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, Sept 2009, pp. 472–477.

[25] P. K. Sahoo, P. Kanungo, and K. Parvathi, "Three frame based adaptive background subtraction," in *2014 International Conference on High Performance Computing and Applications (ICHPCA)*, Dec 2014, pp. 1–5.

[26] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, June 2004, pp. II–302–II–309 Vol.2.

[27] C. Ciliberto, U. Pattacini, L. Natale, F. Nori, and G. Metta, "Reexamining lucas-kanade method for real-time independent motion detection: Application to the icub humanoid robot," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011, pp. 4154–4160.

[28] O. Watanabe, T. Fukuhara, and H. Kiya, "Fast identification of jpeg 2000 images for digital cinema profiles," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 881–884.

[29] C. Y. Lin, Z. Y. Jian, and W. Y. Lin, "Image bit-planes representation for moving object detection in real-time video surveillance," in *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, May 2016, pp. 1–2.

[30] R. Vezzani and R. Cucchiara, "Annotation collection and online performance evaluation for video surveillance: The visor project," in *2008*

IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance, Sept 2008, pp. 227–234.

Chapter 6 – Appendix

6.1 Appendix 1

6.1.1 Proposed algorithm MATLAB code

```
%Extracting frames from video
a=VideoReader('8810.mp4');
for img=1:270;
    filename=strcat('frame',num2str(img),'.jpg');
    b=read(a,img);
    imwrite(b,filename);
end
N = 270;% Number of extracted frames
T=cell(N,9);%bit plane cell
xored = cell(N,8);%cell for xor
final1 = cell(N-1,8); % xored and bit plane sliced cell
grayfinal=cell(N-1,1);% combined cell
%Converting original images to grayscale images
for i=1:N;
    I=imread(strcat('frame',num2str(i),'.jpg'));
    bbb= rgb2gray(I);
    T{i,1}=bbb;
    T{i,1}=double(T{i,1});
    figure(i);
    subplot(3,3,1);
    imshow(bbb); hold on;
    filename=strcat('figure',num2str(i),'.jpg');
    imwrite(bbb,filename);
    % bit plane slicing
    for j=2:9
```

```
26          T{i,j}=mod(floor(T{i,1}/2^(j-2)),2);
27          subplot(3,3,j); imshow(T{i,j});
28      end
29  end
30  % applying xor opeartion
31      for tt=1:N-1
32      for kk=1:8
33          final1{tt,kk}=xor(T{tt,kk+1},T{tt+1,kk+1});
34      end
35      end
36  % combination of xored bit plane sliced images
37  for yr=1:N-1
38      grayfinal{yr,1}=128*final1{yr,8}+64*final1{yr,7}+32*final1{yr,6}+...
39      16*final1{yr,5};
40    grayfinal{yr,1}=uint8(grayfinal{yr,1});
41  % Applying morphological operator
42  B = bwareaopen(A,300);
43  % Memorization when the object stops
44  if (isequal(B,zeros(size(A)))==1)
45      grayfinal{yr,1}=grayfinal{yr-1,1};
46     A=grayfinal{yr,1};
47  %Thresholding
48  t=35;
49  ind_below=(A<t);
50  ind_above=(A>t);
51  A(ind_below)=0;
52  A(ind_above)=255;
53  %Applying morphological opeartor
54  B = bwareaopen(A,300);
55  end
56   resultfile=strcat('proposed',num2str(yr),'.jpg');
57     imwrite(B,resultfile);
58  end
```

6.2 Appendix 2

6.2.1 Background Subtraction algorithm MATLAB code

```
%Extracting frames from video
a=VideoReader('8810.mp4');
for img=95:270;
    filename=strcat('frame',num2str(img),'.jpg');
    b=read(a,img);
    imwrite(b,filename);
end
%Converting original images to grayscale images
for i=1:270;
    I=imread(strcat('frame',num2str(i),'.jpg'));
    bbb= rgb2gray(I);
    T{i,1}=bbb;
    T{i,1}=double(T{i,1});
    figure(i);
    subplot(3,3,1);
    imshow(bbb);
    hold on;
    filename=strcat('figure',num2str(i),'.jpg');
    imwrite(bbb,filename);
end
%Applying background model
background = imread('background model.jpg');
% Calculating absolute difference between current frame and background
% model
for ty=1:270
    current = imread(strcat('figure',num2str(ty),'.jpg'));
    diff = imabsdiff(current,background);
% Thresholding
```

```
t = 35;
% find values below
ind_below = (diff < t);
% find values above
ind_above = (diff >= t);
% set values below to black
diff(ind_below) = 0;
% set values above to white
diff(ind_above) = 255;
%Applying median filter
filtered = medfilt2(diff);
result=strcat('BS',num2str(ty),'.jpg');
   imwrite(filtered,result);
imshow(filtered);
end
```

6.3 Appendix 3

6.3.1 Two-frame differencing algorithm MATLAB code

```
%Extracting frames from video
a=VideoReader('8810.mp4');
%N=number of frames
for img=1:N;
    filename=strcat('frame',num2str(img),'.jpg');
    b=read(a,img);
    imwrite(b,filename);
end
%Converting original images to grayscale images
for i=1:270;
    I=imread(strcat('frame',num2str(i),'.jpg'));
    bbb= rgb2gray(I);
    T{i,1}=bbb;
    T{i,1}=double(T{i,1});
    figure(i);
    subplot(3,3,1);
    imshow(bbb);
    hold on;
    filename=strcat('figure',num2str(i),'.jpg');
    imwrite(bbb,filename);
end
N=270;
% Calculating absolute difference between current frame and background
% model
for tr=1:N-1
    I = imread(strcat('figure',num2str(tr),'.jpg'));
    J = imread(strcat('figure',num2str(tr+1),'.jpg'));
    K=imabsdiff(J,I);
```

```
% Thresholding
 t = 35;
% find values below
ind_below = (K < t);
% find values above
ind_above = (K >= t);
% set values below to black
K(ind_below) = 0;
% set values above to white
K(ind_above) = 255;
%Filtering
B=medfilt2(K);
imshow(B);
newfile = strcat('interframe',num2str(tr),'.jpg');
imwrite(B,newfile);
end
```

6.4 Appendix 4

6.4.1 Three-frame differencing algorithm MATLAB code

```
%Extracting frames from video
a=VideoReader('8810.mp4');
%N=number of frames
for img=1:N;
    filename=strcat('frame',num2str(img),'.jpg');
    b=read(a,img);
    imwrite(b,filename);
end
%Converting original images to grayscale images
for i=1:270;
    I=imread(strcat('frame',num2str(i),'.jpg'));
    bbb= rgb2gray(I);
    T{i,1}=bbb;
    T{i,1}=double(T{i,1});
    figure(i);
    subplot(3,3,1);
    imshow(bbb);
    hold on;
    filename=strcat('figure',num2str(i),'.jpg');
    imwrite(bbb,filename);
end
N=270;
% Calculating absolute difference between current frame and background
% model
for ut=1:N-2
    I = imread(strcat('figure',num2str(ut),'.jpg'));
    K = imread(strcat('figure',num2str(ut+1),'.jpg'));
    L = imread(strcat('figure',num2str(ut+2),'.jpg'));
```

```
    Q = imabsdiff(K,I);
    P = imabsdiff(L,K);
%thresholding
t = 35;
% find values below
ind_below = (Q < t);
% find values above
ind_above = (Q >= t);
% set values below to black
Q(ind_below) = 0;
% set values above to white
Q(ind_above) = 255;
% find values below
ind_below1 = (P < t);
% find values above
ind_above1 = (P >= t);
% set values below to black
P(ind_below1) = 0;
% set values above to white
P(ind_above1) = 255;
R = and(P,Q);
%Filtering
R=medfilt2(R);
imshow(R);
filename=(strcat('threeframe',num2str(ut),'.jpg'));
imwrite(R,filename);
end
```

6.5 Appendix 5

6.5.1 Optical Flow algorithm MATLAB code

```
%Extracting frames from video
a=VideoReader('8810.mp4');
%N=number of frames
for img=1:N;
    filename=strcat('frame',num2str(img),'.jpg');
    b=read(a,img);
    imwrite(b,filename);
end
%Converting original images to grayscale images
for i=1:N;
    I=imread(strcat('frame',num2str(i),'.jpg'));
    bbb= rgb2gray(I);
    T{i,1}=bbb;
    T{i,1}=double(T{i,1});
    figure(i);
    subplot(3,3,1);
    imshow(bbb);
   %hold on;
    filename=strcat('figure',num2str(i),'.jpg');
    imwrite(bbb,filename);
end
%Calculating optical flow
opticalFlow = vision.OpticalFlow('ReferenceFrameSource', 'Input port', ...
    'Method', 'Lucas-Kanade');
N=270;
for op=1:N-1
        IC1 = im2double(imread(strcat('figure',num2str(op),'.jpg')));
    IC2 = im2double(imread(strcat('figure',num2str(op+1),'.jpg')));
```

```
29      opt=step(opticalFlow,IC2,IC1);
30      %Filtering
31      optnew = medfilt2(opt);
32      imshow(opt);
33      optfilename=strcat('optical',num2str(op),'.jpg');
34      imwrite(optnew,optfilename);
35  end
```