Thesis for the Degree of Doctor of Philosophy

NEIGHBOR-BASED INTENSITY CORRECTION: A NOVEL BACKGROUND ESTIMATION ALGORITHM FOR DETECTING FOREGROUND

Huynh The Thien

Department of Computer Science and Engineering
Graduate School
Kyung Hee University
Republic of Korea

November 2017

NEIGHBOR-BASED INTENSITY CORRECTION: A NOVEL BACKGROUND ESTIMATION ALGORITHM FOR DETECTING FOREGROUND

Huynh The Thien

Department of Computer Science and Engineering
Graduate School
Kyung Hee University
Republic of Korea

November 2017

Abstract

Foreground detection is a fundamental issue of image processing and computer vision, which is widely used in visual applications such as object detection and tracking, object segmentation and recognition, video-based activity recognition, intelligent environments and motion analysis. Due to taking a prerequisite role in the general video sequence analysis, the performance in terms of accuracy and processing speed of foreground detection partially affects to the performance of an overall system, for example as the classification rate of activity recognition. Fundamentally, background subtraction, known as foreground detection, aims to detect the moving objects from the difference between the current input frame and the reference frame. A good background subtraction approach is able to robustly work with various hard conditions, such as lighting changes, repetitive object movements, shot- and long-term changes, while maintaining the high-speed processing. In this thesis, we present a robust background subtraction method which is utilized to precisely extract foreground in real-time video sequences.

Existing background subtraction models have been studied with proficient background estimation or background modeling algorithms. Besides updating the background to be adaptive to various background challenges, background subtraction methods should be designed for high-speed processing to be able to realistic systems. However, most current approaches tried to improve the foreground detection accuracy and increase the computational cost of the background modeling algorithms. It is for sure that they cannot be suitable for low-cost systems, where the limitation of computational ability does not allow us to implement more complicated methods. Obviously, it is the urgent requirement of developing a foreground detection method that is featured by low computational cost, less memory consumption, and accurate detection. As the main contribution of the thesis, we propose an efficient background estimation algorithm, namely Neighbor-based

Intensity Correction (NIC), for detection foreground. Developed for low-cost surveillance systems, the algorithm should have be simple and lightweight if compared with existing ones. Instead of using the background model, our algorithm allows to successively update the background reference image for many kinds of object movement adaptively. In the NIC algorithm, we update the background with a novel rule following the selective model, that means, only a number of selected motion pixels are updated instead of all pixels to save the computational cost. The rule is based on comparing two neighboring pixel intensity patterns, constructed from the background and current frame, based on measurement of intensity homogeneity using standard deviation metric. The foreground is segmented by doing the subtraction operation with the updated background image. Although the NIC algorithm is designed to achieve the high-speed processing in cheap-cost systems, it still produces a comparable accuracy of foreground detection.

In the field of background modeling and foreground extracting, the efficiency of background modeling algorithm is evaluated through the foreground detection result by using several standard quantitative metrics. In the experiments, the NIC algorithm is evaluated using the publicly available standard dataset, namely CDNET 2014, that are widely used for the evaluation of foreground detection. For more details, 10 video samples representing indoor and outdoor scenes with various background challenges such as baseline, dynamic background, camera vibration, intermittent object motion, and bad weather condition are utilized for experiments. NIC algorithm is further investigated under various parameter configurations to validate its performance in terms of accuracy and processing speed. From the experimental results, the proposed NIC algorithm is capable of detecting foreground accurately under various hard conditions while maintaining the high processing speed as well. Besides achieving the comparable processing speed result as ~ 9 FPS on 720×480 video, the NIC algorithm outperforms several existing state-of-the-art approaches, including the background reference image based and the background model based, in the terms of accuracy with $\sim 8\%$ F-measure improvement.

Acknowledgements

First and foremost, I gratefully acknowledge Professor Sungyoung Lee, my supervisor, for his fervent encouragement, guidance, and support during my stay in Korea. Besides providing useful advice, he also helps me to sharp research and programming skills to achieve high-quality research publications and further project implementation outcomes. He always generates the best favorable conditions of not only the academic research but also the campus life. He is willing to sponsor the whole traveling for me to participate premium conference in the field.

I also send my profound gratitude to Professor Thuong Le-Tien, my advisor in Master, who has recommended and introduced me to Ubiquitous Computing research group of Professor Sungyoung Lee. He always follows and encourages me in the most difficult moments whenever I was fallen in the mesh of research.

I am grateful to dissertation evaluation committee for highly technical comments and valuable suggestions during the period of the dissertation defense. Besides that, academic recommendations are really useful for me to improve the quality of this dissertation.

I am very thankful to all of my current and former colleagues in Ubiquitous Computing research group their kind supports during my research time at Kyung Hee University. They are not only colleagues but good friends and always ready to help me, especially Korean members. Especially, I show deep gratitude to Professor Oresti Banos and Le Ba Vui, who gave me many technical comments in my publications.

Finally, I am grateful to my parents who are always beside me to share my pain and also congratulate my achievement. Their love is my great motivation to complete this dissertation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Analyzing and understanding image sequences are currently the active researches in image processing and computer vision areas. Many applications in the field such as video surveillance, object recognition, human activity recognition, human-computer interaction, and also multimedia-related applications request moving object detection in a scene as the first step. Obviously, object detection is one of the most important pre-processing steps in many image processing workflows and should be, therefore, developed efficiently because the final system performance in the terms of accuracy and processing speed relatively depends on the outcome of object detection, particularly, this is the foreground image. The basic idea of foreground detection is the separation of an image with the moving object area, called as foreground and denoted by 1-bit pixels, from the static information, called as background and denoted by 0-bit pixels. The most popular approach widely used for the task of foreground-background separation is background subtraction due to the simplicity, in which the foreground image, sometimes called the foreground mask, typically presented in binary, is achieved by the subtraction operation of the input frame and the background reference image or the background model (see the overview of background subtraction approach in Figure 1.1). It is realized that the main task of a background subtraction method is estimating the background image or model without moving objects in a scene. The background is unavailable at the beginning, therefore it should be initialized and updated successively to adapt critical situations likes luminance changes, object appearance, and disappearance of the object in a scene. Most background subtraction methods contribute to background estimation and modeling to improve foreground detection accuracy under various background challenges, but they usually ignore the aspects of processing speed and memory consumption of the algorithm. Many current model-based

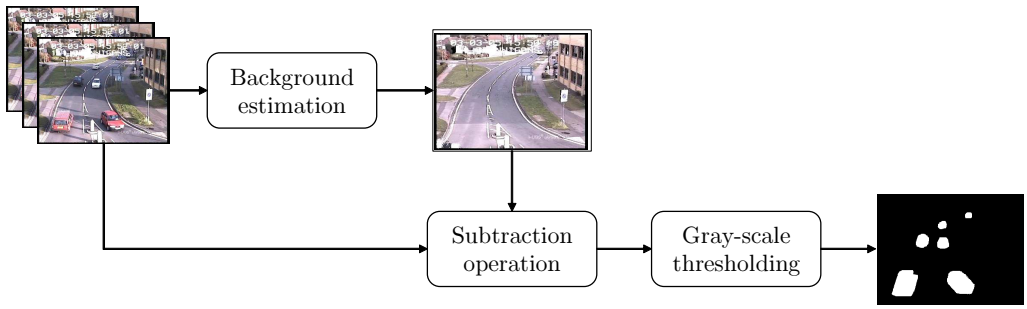


Figure 1.1: Overview of background subtraction approach, where background estimation is the main contribution.

background subtraction methods achieve remarkable results of foreground detection accuracy, but nevertheless, their computational cost is so expensive, usually measure by FPS (frame per second) metric. A general solution commonly considered for low-cost systems, where the limitations of computational ability and memory capacity do not allow the system to process in real-time, is based on the background reference image.

Nowadays, network surveillance systems over internet protocol (see Figure 1.2) have been designed for numerous indoor and outdoor applications including traffic analysis, healthcare, public safety, wildlife tracking, weather monitoring, and etc. These kinds of systems are supported by the rapid growth of technology that allows the surveillance system to work with more private and secure than the analog closed-circuit television (CCTV) systems. Although an enormous amount of research has been academically introduced and industrially applied in practice, the processing and transmission process at each node in the network is still challenging, especially when they are working with real-time demands. In general, each node is equipped with one or more cameras, a microprocessing unit, a storage unit, a transceiver, and a power supply. Besides video capture, the basic functions of each node also include the video processing, coding, and transmission. Some preliminary vision tasks such as moving object detection and localization are able to be done at the sensor node to save the transmission bandwidth whenever the meaningful information such as tracking location is transferred to the network instead of full video. This issue is much more significant for wireless video surveillance system as Figure 1.2(b). Due to working with many cameras at each node, a low-computation background subtraction algorithm to detect moving objects is required to embed on the microprocessing unit. This unit is typically integrated into the

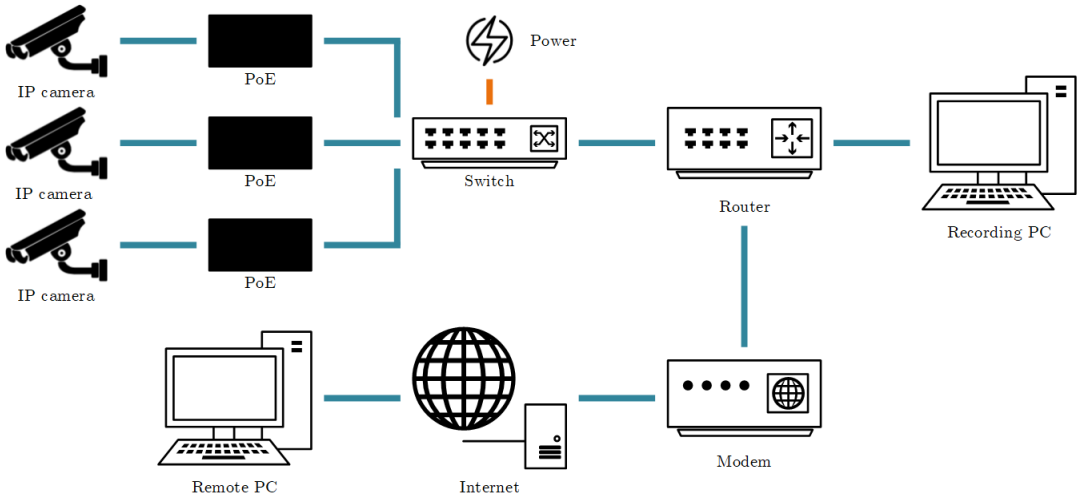
client node with the limitations of computational ability and memory capacity.

1.2 Motivation

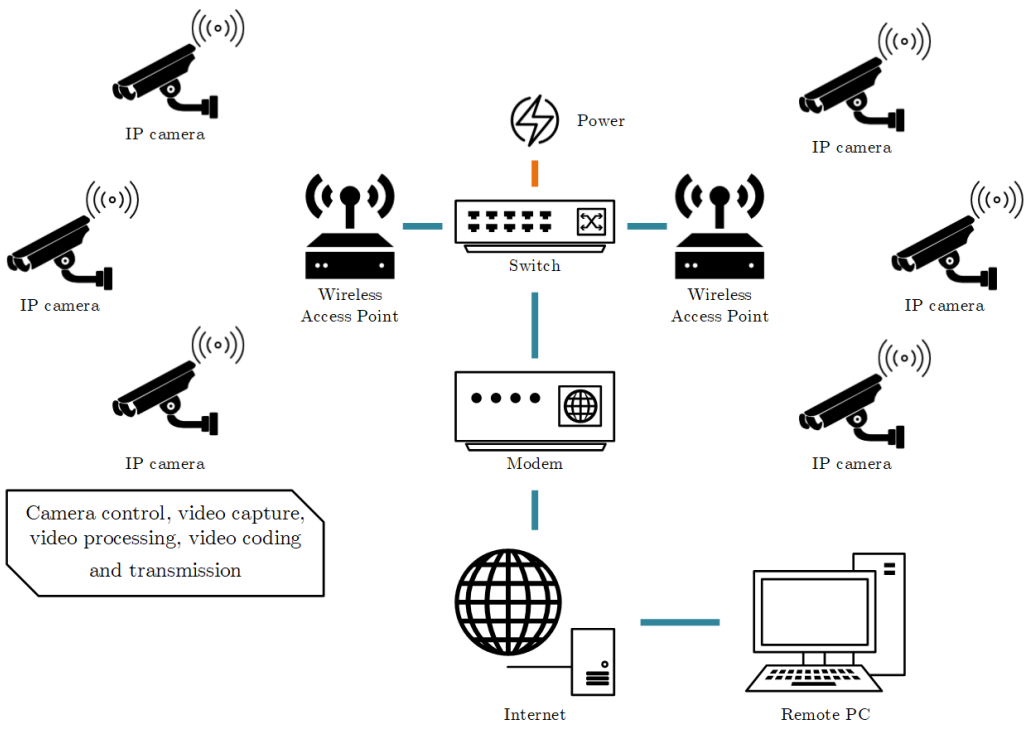
Due to role-playing as the most important preliminary step in many image processing and computer vision applications, the requirements of foreground detection for accuracy and processing speed are very high to guarantee the performance of the overall systems, for example, the accuracy of multi-objects detection and tracking. Therefore, a background subtraction algorithm should be designed to effectively work with various background challenges in the real world such as baseline, dynamic background, camera jitter, intermittent object motion, bad weather, and etc. On the other hand, the processing speed of an algorithm is truly important to realistic systems, therefore, the computational cost should be optimized for suitable to cheap cost devices as IP cameras (see Figure 1.3), wherein a power-saving central processing unit (CPU) with is integrated inside. With the proliferation of IP networks and falling prices on IP video cameras, IP video surveillance becomes an attractive solution for many industry sectors. Compared with two older solutions including the analog CCTV system and the analog camera with video server on an IP network, the network camera-based video system is introduced. With the sequentially upgraded infrastructure of the Internet, IP cameras have been widely used in video surveillance systems thanks to numerous advantages including superior image quality, cost-effectiveness, scalability and flexibility, distributed intelligence, remote accessibility, easy and reliable installation.

Easy, future-proof integration It is obviously to believe that a video surveillance network using IP camera can be developed in any places for various different purposes. The IP-based network is potential to combine with other equipments for high-advanced functions as an integration system. For a vast amount of practical applications, a fully unified video-based surveillance network is built for access control, management, and warning. While analog systems are generally restricted in the expansion, integration, and upgrade.

Superior image quality Image quality is recognized as one of the most important features of any video-based systems. Superior image quality providing more details in images allows users or customers making decisions accurately and quickly for more effective people and



(a)



(b)

Figure 1.2: Video surveillance systems using IP camera network: (a) the wired network, (b) the wireless network.

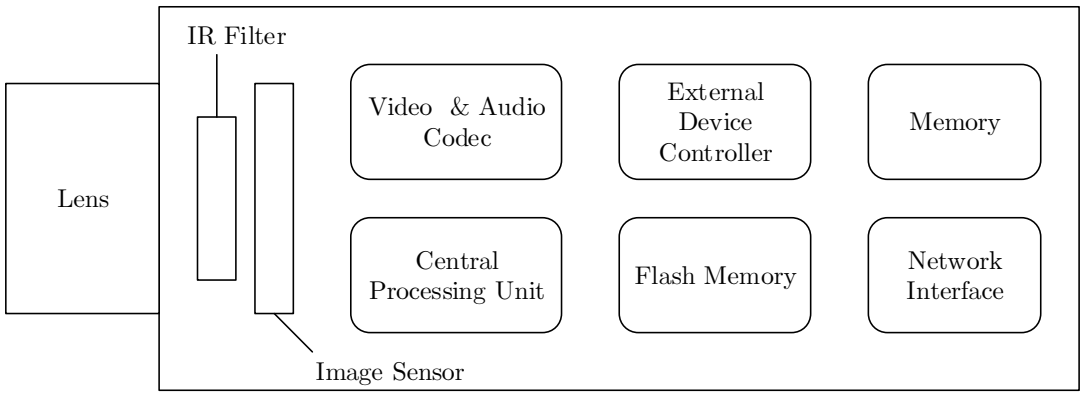


Figure 1.3: General components of a IP camera.

property protection. A high-quality video system enables the image or video processing algorithms, embedded into IP cameras, to achieve higher accuracy rates based on processing much more image details. In the most of current IP surveillance systems, images from a network camera are converted to digital for transmission without any degradations of quality. Additionally, the ability of data access from the user side is much more easily and conveniently if compared with analog systems.

Cost-effectiveness Video products and add-on equipments for IP networks operate on open standards. It can be seen that using standard hardware as personal computing server rather than proprietary apparatus or device such Digital Video Recorder (DVR) helps to optimize management and equipment costs significantly, particularly for big systems where storage devices and servers are so expensive. Additionally, cost saving also comes from infrastructure and maintenance. IP-based networks, for example, Local Area Network (LAN) and the Internet, are shared and exploited for many other applications. It is reasonable to believe that a network video system somehow gives insights into ways of enhancing commercial activity.

Scalability and flexibility A network video system is mostly developed to adapt user requirements. Clearly, IP-based systems including network hardware as video devices and network software as applications are designed to operate fluently on a shared wired or wireless network. For data transmission, many kinds of data including video, audio, I/O command, power, and other data are carried by only one same cable. With Power over Ethernet (PoE),

the Ethernet cables can supply power to network devices as IP camera by a data connection. Any number of network video devices can be easily added to the existing system without any significant changes of the network. Network video devices can be set up for almost any locations, and the system can be controlled and managed over the IP network. Obviously, development of a network video system by using standard devices and protocols has many benefits from the growth of technology. For example, video data can be stored on redundant servers which are placed in different locations over the world for reliability increment.

Distributed intelligence Due to lack of time, so many recorded videos are never watched or quickly reviewed by supervisors at the monitoring center of a video surveillance system. As a result, events and human activities are ignored without any notices. A solution for human-free monitoring in these systems is embedding intelligent algorithms into the camera itself. Nowadays, IP cameras and video encoders are integrated several built-in features such as moving object detection and tracking for suspicious behavior recognition. These intelligent features allows network cameras constantly process inputs and quickly response outputs to supervisors by visual or sound warnings. Usually, some popular features as people counting, vehicle tracking, and abnormal behavior detection are developed as third-party applications which can be downloaded and installed for IP camera devices.

Remote accessibility From any authorized client computers, users can access videos over the network in the real-time at any time and anywhere. IP-based video products and devices provide an easy manner to distribute high-quality videos over LAN or the Internet. The recorded videos are usually stored at monitoring centers to guarantee convenience and security while the information can be transmitted over the network.

Easy, reliable installation The whole installation of an IP surveillance system is done easily and quickly. As a remarkable feature, IP cameras can be set up with only one cable for data transmission and power supply, that is offered by PoE standardization. Some kinds of camera have been designed and manufactured for specific toughest conditions such as outdoor and bad weather.

At the present time, a massive amount video is being recorded without watching or analysis

Table 1.1: Processing speed summarized from http://changedetection.net/

Method	Facility	Processing Speed (FPS)
GMM [1999]	C++ on Core i7 3.4GHz	$\sim$21 FPS on 720×480 video
KDE [2000]	C++ on Core i7 3.4GHz	$\sim$9 FPS on 720×480 video
EGMM [2004]	C++ on Core i7 3.4GHz	$\sim$49 FPS on 720×480 video
SGMM-SOD [2011]	C++ on AMD Phenom II X4 3.0GHz	$\sim$34 FPS on 320×240 video
SGMM [2012]	C++ on AMD Phenom II X4 3.0GHz	$\sim$14 FPS on 720×480 video
DPGMM [2012]	OpenCL on GeForce GTX 580	$\sim$28.5 FPS on 320×240 video
PBAS [2012]	C++ on Core i7 3.5GHz	$\sim$48 FPS (average of different resolutions)
STMB [2014]	C++/CUDA on Core i7 2.3GHz	$\sim$12 FPS on 320×240 video
SharedModel [2015]	C++ on Core i7 laptop	$\sim$35 FPS on 320×240 video
PAWCS [2015]	C++ on Core i5	$\sim$27 FPS (average of different resolutions)
SuBSENSE [2015]	C++ on Core i5	$\sim$45 FPS (average of different resolutions)
FTSG [2014]	Matlab on Core 2 Duo 2.4GHz laptop	$\sim$10 FPS on 320×240 video
M4CD v2.0 [2016]	Matlab on Core i5-3210M laptop	$\sim$0.2 FPS on 320×240 video
Simp-SOBS [2017]	Matlab on Core i7 2.3GHz laptop	$\sim$0.06 FPS on 720×576 video
SBBS [2017]	Matlab on Xeon desktop	$\sim$10 FPS

from human due to lack of time. Clearly, events and activities are missed and some abnormal behaviors, therefore, remains unnoticed. With IP camera, in fact, is a small self-contained computer with several built-in features such as motion detection, object tracking, active tampering alarm, and etc. Among them, motion detection is a regular function performed by the CPU integrated inside the IP camera. Besides that, some higher-level video analysis functions are also embedded such as object tracking and abnormal behavior. Obviously, performing video processing algorithms on a low-cost computation CPU is really challenging, that requires algorithm should be optimized for processing speed before embedding.

Therefore, the motivation is to make the use of advantages of IP cameras for video surveillance systems, in which intelligent features are developed from video processing algorithms and embedded inside the camera. We propose an efficient background subtraction algorithm to detect moving objects, which is considered as a regular function, with the highlights including low computational and memory consumption while maintaining a comparable accuracy. The proposed algorithm is designed to be suitable for not only IP cameras but also cheap-cost devices such as Raspberry Pi and Arduino, in which the computational ability of the processing unit and the capacity of the memory are limited.

1.3 Problem Statement

Although state-of-the-art background subtraction approaches based on modeling the background reference image essentially achieve remarkable outcomes in the field of foreground detection, some challenging issues need to be addressed in this work:

- **Lack of an accurate background updating scheme in background reference image based approaches.** Background maintenance aims to successively update the background reference image or background model based on the information of input frames, previous backgrounds, and foreground mask even. Since a *pure* static background does not exist in the real world, an updating scheme is, therefore, necessary for learning the changes occurred in a scene over time likes the change of daylight, background motion, intermittent object movement, and so on. In most traditional methods, an initial background image is basically estimated from a buffer which is the collection of a number of input frames. For example, average filtering approach calculates the background image as the average values of N previous frames gathered into the buffer. In order to update background image, the filtering approaches such as average or median gather the current input frame to the buffer and simultaneously eliminate the oldest one from the buffer. The quality of estimated background image depends on the size of the buffer and also the buffer sampling rate. Typically, a larger buffer is better for more accurate of background modeling, however, the demand of memory and computation are rapidly increased. In running average approach, the updating is performed based on the previous background and input frame with a learning rate, which is called as recursive updating scheme. Several other schemes consecutively update the background image based on mean and variance values of previous backgrounds and histogram information. A typical drawback of current schemes is that a fixed learning rate is used for updating all pixels over time. Although many improvement versions have been introduced with an adaptive learning parameter, they are fragile to multi-modal background challenge. Additionally, they ignore the spatial information of pixel intensity in the updating progress, which sometimes conducts inaccurate estimation.

- **Huge memory consuming and expensive computation cost for updating background**

over time. Compared with detection accuracy, processing speed is not considered and evaluated thoughtfully in many background subtraction methods. However, this is the most important criteria whenever we implement a background subtraction algorithm for any realistic systems. Besides that, we usually trade computation cost for improvement of accuracy during the algorithm development progress. Obviously, in order to upgrade the accuracy of foreground detection, especially in various highly challenging backgrounds, so many techniques and processing steps have been combined and included in the updating progress, that can increase the complexity, computation cost, and memory requirement. For example, in the filtering approaches, the quality of background image can be improved by enlarging the buffer size, but nevertheless, the system will require more memory for bigger buffer and more computation resource for estimating the background. In model-based background subtraction methods, the buffer is replaced by the background model that needs to be stored. This drawback is addressed by the running average scheme for updating, however, the accuracy cannot be maintained if compared with filtering approaches. Clearly, achieving a good balance of accuracy and processing speed is not an easy task, especially when working on low-cost devices.

Based on addressing two challenging issues of most current background subtraction approaches, the thesis focuses on developing an image-based background subtraction method which has an efficient scheme for background updating. The proposed algorithm is expected to improve the foreground detection accuracy with a low computation cost if compared with existing background image-based methods.

1.4 Proposed Concept

In this work, we propose a novel background estimation algorithm, namely Neighbor-based Intensity Correction (NIC), for the purpose of foreground detection. The proposed algorithm is developed from the idea of replacing object intensity by background intensity. By this way, the object in the background image is removed over time. With the fundamental assumption that the background class is more homogeneous than the object class, the intensity replacement is done based

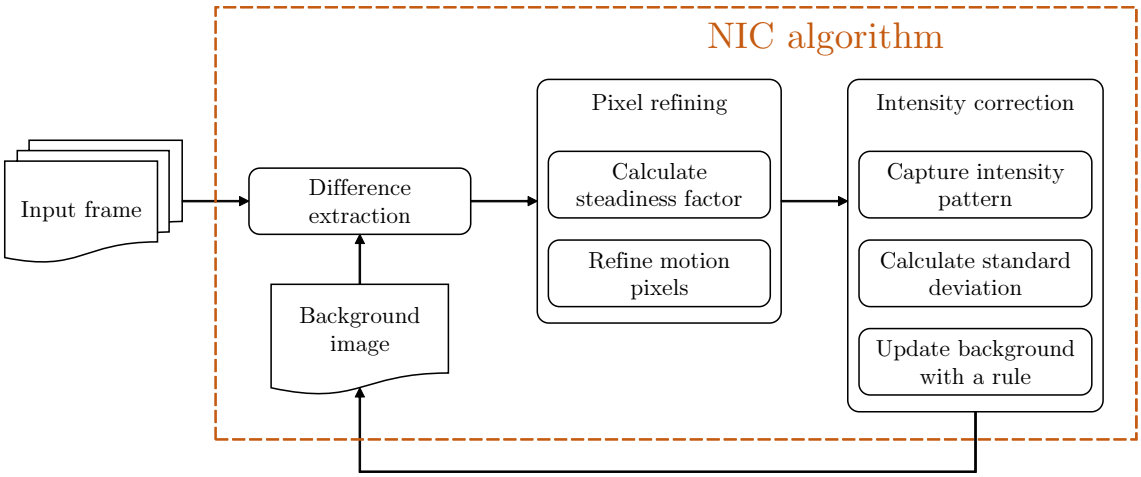


Figure 1.4: The general workflow of the background estimation approach with NIC algorithm (difference extraction is commonly used in several algorithms as a preprocessing step).

on analysis of the homogeneity of pixel intensity in the background image and also in the input frame. This task is completed by a proposed intensity updating rule. Consideration of neighbor pixels surrounding each motion pixels to analyze the homogeneity of a pattern is the recipe for updating background image accurately. For more explanations, a higher homogeneity of a pattern has a smaller standard deviation of intensity values. Thus, in the updating rule, a motion pixel at the background image will be updated to the pattern whose standard deviation is less than another. Furthermore, we also propose a factor which is capable of detecting infrequent background motion pixels as outliers, called steadiness factor. Fundamentally, this factor is updated at each input frame for all pixels via a steadiness matrix. The benefits of outliers elimination includes maintaining the quality of estimated background image and reducing the computational cost during the updating progress. In summary, our proposed NIC algorithm includes two steps: pixel refining has the role of outliers elimination using the steadiness factor and intensity correction contributes as a background maintenance scheme with an intensity updating rule. The general workflow of the background estimation approach is presented in Figure 1.4 and detail described in Chapter 3.

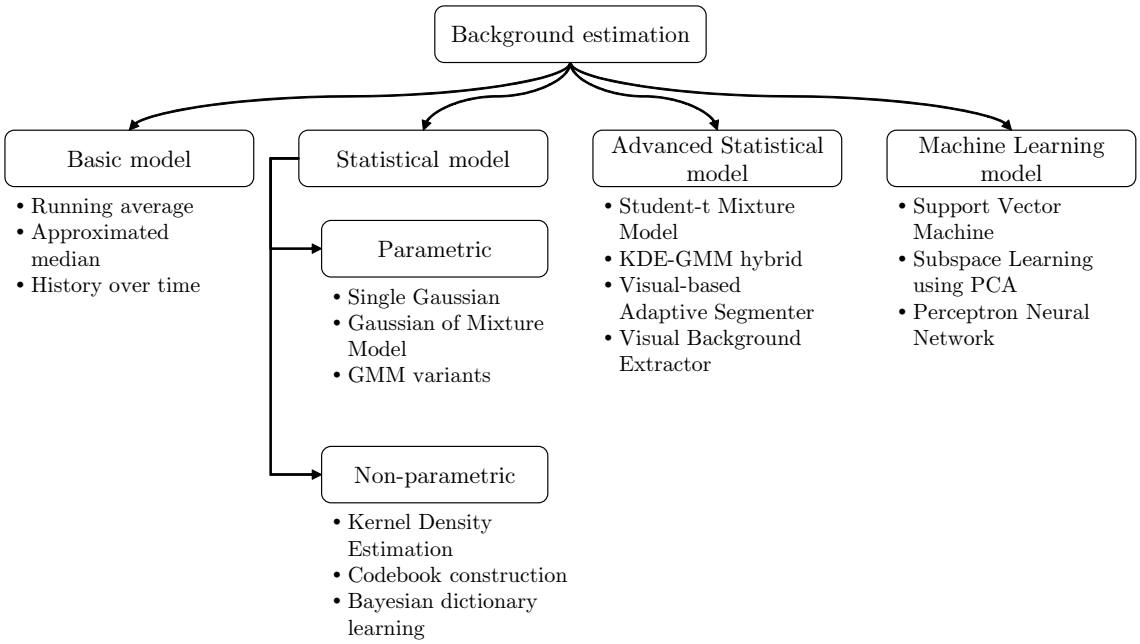


Figure 1.5: Overview of background subtraction approach, where background estimation is the main contribution.

1.5 Key Contributions

From the algorithm summarization in Figure 1.5, the background of a scene is fundamentally modeled by statistical approaches to be against illumination change and background motion. From a simple idea, the probability density function of pixel intensity is modeled by a single Gaussian. A pixel from an input image is considered as a background pixel if its intensity value is appropriately described by its density function. Nevertheless, a single Gaussian model is unable to address the dynamic background challenge fluently. Gaussian Mixture Model (GMM) is, therefore, developed to model a multi-modal background image sequence based on two assumptions: (i) the background region should be larger the foreground region and (ii) the background intensity of each pixel is more frequently visible than the foreground intensity representing moving object. Although GMM yields a quite impressive performance of foreground detection, it is unsuitable for hardware implementation due to the requirement of parameter estimation for each of particular background scenario. To overcome the nature drawback of GMM and also its variants, a non-parameter approach, namely Kernel Density Estimation (KDE), is proposed to estimate back-

ground probabilities from several recent samples over time. However, KDE requests much more computational cost and memory resource for the task of pixel-wise kernel estimating. As a result, KDE detects foreground slower than GMM, i.e. processing speed measured by FPS of KDE is lower than those of GMM as the results reported in Table 1.1. A vast amount of advanced statistical model improves the accuracy of foreground detection, however, most of them reserve large memory for storing background model and request expensive computation cost for updating the model over time. Besides that, some of the background models are quite complicated to unsuitably implement for realistic systems. It is observed that the background models represented in Table 1.1 are benchmarked in term of processing speed using C++ on the powerful computers, however, the FPS results are not impressive. More simple than background model approaches, traditional background subtraction methods estimate the background as a reference image. Some baseline algorithms in this group are average/median filtering, average running, approximated median, running Gaussian average, and histogram over time. With filtering approaches and histogram over time, the background image is estimated based on the observation of several input frames, that means, several input frames stored in a buffer have been used for calculating the background as the mean or the median of pixel values. The background reference image is updated consecutively for each input frame by updating the buffer. It is noted that the buffer has a fixed size. These methods are much more memory consuming for the buffer installation and exhaust computational resource for estimating the background over time. The processing speed can be increased for them by reducing the buffer sampling rate. On the other hand, some quickly updating schemes have been recommended, in which the background image is updated based on the input frame with a learning rate. Unlike the model-based background subtraction methods, the reference image based methods are typically fragile under illumination change and background motion. Additionally, they are unavailable in working with various challenging backgrounds due to the use of fixed learning rate for updating pixel intensity over time. However, for some middle challenges, they are considered to embed in realistic systems due to its low-cost computation.

The dissertation focuses on providing a solution for major challenges relating to foreground detection using the background subtraction approach. In particular, the major contribution is Neighbor-based Intensity Correction (NIC) algorithm, a novel background estimation algorithm,

that is able to work robustly in several common background scenarios. The algorithm is designed to detect moving objects in a scene with two highlighting features including low computational cost and less memory consumption. Besides that, the algorithm is also simple to be much more suitable for low-cost visual surveillance systems, especially IP camera network. In details, NIC algorithm includes two major concepts: (i) an efficient background updating scheme for accurate foreground detection and less memory in use, and (ii) a novel factor for boosting processing speed.

- We propose a novel factor, called as steadiness factor, which is able to measure the variation frequency of pixel intensity between the input frame and the background image. This factor is calculated for all pixels based on the result of difference extraction and then is kept in a steadiness matrix. The steadiness factor is used to discard infrequent motion pixels which are not considered for updating. Obviously, the benefits expected from capably controlling outliers are accuracy preservation and computation minimization.

- We further propose a novel background maintenance scheme for updating the background image sequentially at each input frame. Instead of going forward with a background model, we use a reference image for representing the background of a scene as a solution to save memory. The scheme aims to update the intensity of object pixels by the intensity of background pixels with a proposed intensity updating rule. The rule is done based on the analysis and comparison of the intensity pattern captured in the current background image and another one in the input frame. The standard deviation results of intensity values of two corresponding patterns are compared for making a decision of intensity correction. Since the pattern acquires not only the object pixel but also the neighboring pixels, quality of updating progress is remarkably improved.

In order to demonstrate the efficiency of our proposed background estimation, experimental results on various videos, which represent several background challenge in the field of change detection, are reported. We further benchmark the algorithm under various parameter configurations to select the best reasonable setting as a default. Our proposed algorithm achieves the remarkable performance in terms of foreground detection accuracy with 0.8191 of F-measure and processing speed with $\sim$9fps.

1.6 Thesis Organization

This dissertation is organized into chapters as following.

- **Chapter 1: Introduction**. Chapter 1 provides a brief introduction to the research work of background estimation. The chapter further summarized several major problems in the area of background estimation and the limitations of current approaches. After that, the goal and the overview of the contribution of the dissertation are presented.

- **Chapter 2: Related work**. Chapter 2 provides the detail review and discussion on previous approaches of background subtraction for foreground detection and their corresponding limitations. In addition, the remarkable applications of background estimation in video-based systems are also given.

- **Chapter 3: Proposed methodology for foreground detection**. In this chapter, we present a novel background estimation algorithm, namely Neighbor-based Intensity Correction (NIC), which is utilized in background subtraction to detect foreground. We mainly propose an efficient background maintenance scheme which allows to precisely estimate the background image with a proposed intensity updating rule, in which the intensity patterns of the background image and the input frame are analyzed based on the standard deviation metric with an updating rule.

- **Chapter 4: Experimental results and discussions**. We provide the description of testing datasets, currently available publicity, which are utilized for evaluating the performance of our proposed background estimation algorithm on results of foreground detection The performance is benchmarked based on seven standard metrics in the field. Furthermore, the experiment results are compared to other state-of-the-art approaches of background reference image based and background model based to prove the remarkable performance of NIC algorithm.

- **Chapter 6: Conclusion and future directions**. This chapter concludes the dissertation with some discussions of limitations and also provides future directions for performance improvement.

Chapter 2

Related Work

This chapter presents state-of-the-art background subtraction methods for detecting foreground in a scene. Totally, four major categories of methods contributed to background subtraction are the basic models, the statistical models which comprise of parametric and non-parametric, the advanced statistical models, and machine learning models. The highlight and limitation of current algorithms are discussed particularly. Additionally, the application of background subtraction algorithm for moving object detection and tracking in video-based systems are given in this chapter.

2.1 Background Subtraction for Moving Object Detection

A video is gathered by a number of static images or frames with audio data. The frames in a video, typically recorded by a single camera, are directly utilized for analyzing the visual content in surveillance systems. After collecting frames by a camera and storing them in a memory, several digital video processing algorithms are applied as the next following steps for detecting moving objects in a scene. Fundamentally, object detection, wherein the foreground representing moving objects is extracted from a scene background, is a preliminary step in many image processing and computer vision tasks such as object tracking. Because monitoring objects in a scene is basically a boring and exhausting assignment in the remote center of surveillance systems, it is necessary to study an automatic object detection and tracking technique. A vast amount of techniques have been studied for the last decade, in which the image features such as color, shape, texture, contours, and motion (i.e., trajectory and spatial relationship) are used for tracking moving objects.

Moving object detection and tracking are popular functions of many intelligent visual surveillance systems. For example, the occupancy detection in the parking area includes an algorithm which is developed to detect and track vehicles [1]. Instead of using individual sensors for each

parking slot as an expensive and unscalable solution, an intelligent parking management system, based on video processing algorithms which are capable of detecting and tracking vehicles, becomes a realistic solution. In order to adapt a huge number of personal vehicles, developing a scalable urban parking system is very important and imperative, especially for crowded cities. These parking systems can help users to find the closest available free parking slot with a trajectory guidance to save time and money. Moreover, the system can provide the warning of restricted areas for non-allowed specific vehicles.

On the other hand, moving object detection and tracking are also useful for calculating statistics of the traffic capacity in road [2], airport [3]. Although many approaches have been introduced for traffic flow surveillance by combining moving object detection and tracking with other technologies as electronic tags installed in vehicles or the Global Positioning System (GPS), for vehicle counting and re-identification, they are poor operating in bad weather and quite expensive. Therefore, an accurate moving object detection and tracking algorithm for traffic surveillance systems should be designed with following features: low computational cost for real-time processing and robustly working in realistic challenging background scenarios. The interesting objects of detection and tracking are diverse such as people [4, 5], animals [6], vehicles [7], airplanes, and boats [8] which are observed in the indoor [9] and outdoor [10] environments. Moreover, several algorithms in the field have been designed to well working with many image types as grayscale [11], color red-green-blue (RGB) [12], depth [13] images, and several camera configurations as stationary, moving, and pan/tilt/zoom (PTZ) [14] cameras. Compared with other techniques, background subtraction is recognized as the most suitable choice for video-based object detection and tracking systems. Currently, background subtraction is still an active research due to many common challenges relating to background modeling: gradual and sudden illumination changes, dynamic background, camouflage, shadows, bootstrapping, video noise, camera jitter, camera automatic adjustment, and intermittent object motion. In general, background subtraction comprises two main stages: the background image/model initialization and updating, and the subtraction between the current frame and the background image/model. Most of the existing methods have contributed on the background model generation and updating schemes.

2.2 Background Subtraction Methodology Review

2.2.1 Basic Model

Average-Running Average

As the simplest approach to extract the foreground mask is subtracting the input frame with moving objects in a scene to a background reference image. The background image can be estimated as the average of previous N input frames [15] which are stored in a buffer. At each pixel, the average values are calculated for all three color channels, respectively. In order to extract the foreground mask, the difference image between the input frame and the estimated background image is compared to a threshold. The background is memory consuming for storing N frames, hence, running average approach [16] is introduced to overcome the current drawback of memory in use. For each input frame, the background reference image is updated with a learning rate, the parameter reflects how much the input frame contributes to the background and also decides how fast the background responses to the change in a scene. Obviously, running average is more efficient than filtering approaches in memory consuming and computation cost. However, a fixed learning rate cannot adapt to dynamic motions in a scene, it should be, therefore, estimated as an adaptive value based on the detected foreground [17]. In [18], the spectral, spatial, and temporal features are combined to identify the most accurate background candidates for modeling. Another drawback of filtering and running average approaches is only one global threshold for all pixels in the image.

Median-Approximated Median

Besides the average filtering, median filtering is commonly used to estimate the background reference image, in which the value of each pixel is equal the median value of that pixel from previous N input frames stored in a buffer. Another more efficient method with less memory in use is the approximated median [19]. In the background reference image, the value of a pixel is increased by 1 if the input value is larger than the previous estimation, and decreased by 1 if the input value is smaller than previous one. In [20], the deviation between the median and its neighbors in a sequence of re-ordered pixels is calculated to partition the background data for estimating

the background reference image. The proposed median model achieves a remarkable efficiency in the case of slow moving and stationary vehicles, however, the complexity is quite expensive for real-time applications. The approximated median model is improved in [21] by a pre-defined parameter which adaptively allows determining the speed of updating. A fast version of temporal median model [22] contains a simple mechanism for median checking of two consecutive frames based on high correlation characteristic. The experimental results prove that the algorithm was two times faster than the histogram selection approach and met the requirement of real-time processing in video-based surveillance systems. The main problem of approximated median approaches is the slow adaptation to background changes.

Histogram Overtime

The histogram-wise technique [23, 24] is developed to deal with illuminance changes in background. Moreover, the proposed model is more robust to dynamic background challenge thanks to identifying stationary objects. An efficient local dependency descriptor, called local dependency histogram (LDH) [25], which is calculated over the area surrounding a pixel to extract its spatial dependency statistic for modeling the background image. Foreground segmentation is done by comparing the LDH of input frame with the weighted LDH of the background image. A local histogram of figure/ground segmentation [26] is recommended to be against to the dynamic background challenge. Concretely, the background image is modeled as a group of adaptive local histograms of figure/ground partitions for representation of the structural characteristics of surrounding area. Extracting the correlation of image variations at every neighbouring pixel allows achieving high accuracy of foreground detection robustly under dynamic scenes. An efficient background model is introduced in [27] using equi-depth histograms to depict multi-dimensional distributions. The model has an ability for real-time applications thanks to very little memory utilization. In [28], a background model based on the intensity histogram of color channels, namely Histogram Min-Max Bucket (HMMB), is built from the minimum and maximum values of non-zero frequencies and was adaptively updated using sliding window scheme. The proposed method yields remarkable performance of foreground segmentation, however, it requests a large number of initial frames and uses expensive memory for training.

Although basic models are mostly simple and easy for implementation, they have some essential limitations: (i) sensitivity with sudden changes of illumination, (ii) non-adaptive working with a multi-modal background, and (iii) expensive cost of background modeling. Therefore, compared to existing state-of-the-art methods, basic models should be significantly improved in terms of robustness, accuracy, and complexity.

2.2.2 Statistical Model

Statistical background model-based approaches can be fundamentally categorized to the parametric and non-parametric group. The parametric group includes methods which are developed based on Single Gaussian, Gaussian of Mixture Model (GMM), and GMM improvements. To overcome the parametric tuning for the probabilistic models, non-parametric approaches such as Kernel Density Estimation (KDE), codebook construction, and dictionary learning are studied for modeling background.

Parametric Model

A simple manner to represent the background is based on the assumption of that the intensity values of pixels over time can be modeled by a single Gaussian [29, 30]. The conditional distribution of the normalized statistical features of several background frames is assumed as a Gaussian distribution [31] to model a better reference background image. By analyzing object motion in the spatial-temporal domain of the input frames and then modeling motion features into a Gaussian model, a background updating scheme [32] is performed to handle such several issues as the artifact, camera shaking, abrupt luminance change. To address the local and global illuminance changes in the indoor and outdoor scenarios, the background is modeled using a uni-modal Gaussian distribution [33] with a quickly adaptive learning rate. The appearance of background is described as a Gaussian component [34] whose model parameters are consecutively updated for each input frame to cope with the luminance changes over time. The method shows an impressive object tracking accuracy, but nevertheless, it is time-consuming for background updating progress. In [35], the background reference image is constructed using one Gaussian per pixel and then updated following the Running Gaussian average model [29]. The foreground-background

classification is done by a graph cut algorithm that allows analyzing neighboring information for boosting the tracking accuracy.

A uni-modal model with a single Gaussian cannot deal with dynamic backgrounds that have background motions as waving tree, swell or water fountain [36]. To overcome this problem, the Gaussian Mixture Model (GMM) [37, 38] or Mixture of Gaussian (MOG) is introduced for the dynamic background challenge. Instead of modeling the values of all pixels by one particular distribution, Stauffer and Grimson models the value of a pixel by a mixture of Gaussians. Based on the variance of each Gaussian in the mixture, the Gaussians corresponding to background class are determined efficiently. Pixel values that do not belong to the background distribution will be considered as foreground class until there is a Gaussian covering them fully.

Many GMM-based improvements are developed to be more accurate, robust, and adaptive to common background challenges in the real world. A general framework of Time-Adaptive, Per-Pixel Mixture of Gaussians (TAPP-MOGs) [39] is developed to incorporate corrective guidance with positive and negative feedback from analysis of image regions, image frames, or object semantics. The typical drawback of GMM is how to identify the number of Gaussian components to be adaptive to the illuminance change in the background image. In [40, 41], GMM parameters are constantly learned by recursive equations for particularly select the appropriate number of components for each pixel. These adaptive algorithms not only boost the processing time but also enhance the segmentation accuracy of foreground detection. Those above methods achieve the benefit of computational time reduction compared to the original GMM.

Some approaches update GMM in other ways to handle several particular problems. A two-layer GMM [42] is developed for detecting moving in dynamic scenes, in which the first layer handles gradually illuminance changes and the second layer deals with suddenly and irregularly pixel intensity changes. This approach has a disadvantage of expensive complexity due to simultaneously estimating GMM parameters of two layers. By considering the variation of intensity of a pixel and its neighboring region, the problem of unstable background is addressed through a spatial-temporal GMM (STGMM) [43]. A combination of pixel-wise and region-wise GMM-based background models in a hierarchical structure [44] allows capturing the spatial relations of surrounding pixels to successfully detect objects in very harsh conditions. To deal with the camera

fitter challenge, Histograms of Oriented Gradients (HOG) is applied to GMM in [45] for evaluating stabilization by working with local image gradients instead of illuminance values. In [46], GMM is extended for modeling the background image in a Bayer-pattern domain and used to segment the foreground by an interpolation scheme of red, green, and blue (RGB).

Non-parametric Model

Although GMM and GMM variants are designed to fit with many kinds of background, they present several disadvantages. A background having quick variations of intensity cannot be precisely modeled using a few Gaussians and the estimation of GMM parameters has a risk of complexity expanse, especially this estimation stage is not easy to implement in real systems [47]. A non-parametric model [48] is studied to estimate the background probabilities at every pixel from a number of recent samples using Kernel Density Estimation (KDE). Since KDE has a limitation of time-consuming, several improvements have been recommended. Through a cooperative domain-range representation of all pixels in an image, the background image is directly modeled using KDE [49]. This model is able to present the multi-modal spatial uncertainties and complex probability density between location and color information. In [50], a general multi-variate KDE model is capable of modeling the background image accurately and also capturing color dependencies without color based probability density. Although the gradual change and sudden change of illuminance in a scene are successfully handled by temporal adaption in the proposed approach, it needs to be optimized for speed and tracking stableness. The major limitation of KDE-based background modeling approaches is huge memory consumption for kernel estimation through a large number of historical observations.

Other non-parametric models considered for background modeling include codebook construction and dictionary learning techniques. In [51], the background image is modeled using the codebook construction technique, in which each pixel is represented by one or more codewords. Sample background values are encoded into a codebook, typically referred as a set of unique codewords, based on the color distortion metric, so the compressed form of background model for a long-term image sequence is characterized with little memory requirement. A sparse representation over dictionary learning technique is developed for background subtraction in [52]. The basic

assumption of this technique is that the current input frame can be summarized as the linear combination of dictionary vectors. The general drawback of codebook construction based and dictionary learning based approaches is huge memory consuming.

2.2.3 Advanced Statistical Model

Recently, several advanced statistical background models have been developed by integrating some innovative techniques and can be categorized into following classes: mixture models, hybrid models, and advanced models.

Mixture Model

Student's t-mixture distribution is optimally exploited for the task of background modeling [53]. This mixture model is much more robust against outliers as noises than GMM due to its more heavily-tailed characteristic. Besides working lightly in real-time systems, the model further yields an impressive foreground-background segmentation accuracy with an online recursive filter scheme for updating parameters of Student's t-distribution. Dirichlet process Gaussian Mixture Models (DP-GMM) [54,55] is capable of truly estimating per-pixel background distributions. Each mode in DP-GMMs has the ability to represent a range of colors using a Gaussian distribution, to learn it from a data stream, and to update it on the changes of a scene over time. DP-GMMs smoothly fits various background challenges by tuning mode adaptively. To overcome the issue of model scalability, the model works with Gibbs sampling only once for each sample and fits with the GPU-based implementation for real-time performance.

Hybrid Model

To approximate the color distribution of the background image accurately, a novel mixture of the non-parametric regional model (KDE) and the parametric pixel-wise model (GMM) is recommended [56]. In this mixture approach, the locality distributions of foreground and background are estimated by KDE for error reduction, and the color, locality, temporal coherence, and spatial consistency are united and learned by a Markov Random Field (MRF) framework. In [57, 58], the probability density function is formulated for modeling background and moving objects by

a KDE-GMM hybrid model (KGHM). By capturing the spatial dependencies of color information of neighboring pixels explicitly, KGHM efficiently works with highly dynamic background challenges.

Advanced Model

Recently, Visual Background Extractor (ViBE) [59–61], is introduced, in which the background model is built by gathering recently observed values at each pixel location and smoothed in decaying lifespan by a random selection policy. ViBE is capable of determining whether a pixel belongs to the background or foreground class based on a random comparison between its intensity and surrounding neighbors. It circulates the pixel intensity to the background model as the neighboring pixel for the next evaluation. Compared with several state-of-the-art methods, ViBE yields an impressive performance in terms of accuracy and computational cost of background estimation and foreground extraction. Furthermore, the authors developed a downscale version for cheap computation in digital cameras. However, it should be improved for highly challenging background models such as dark background, shadows, and frequent background changes. Graph cut optimization [62] is applied to the results of improved ViBE model to boost the foreground segmentation accuracy. An improvement of ViBE for ghost suppression by detecting and re-initializing the region of ghost area is carefully described in [63]. Pixel-Based Adaptive Segmenter (PBAS) [64] models the background from observed pixel values in the temporal domain and updates frame-by-frame with per-pixel learning parameters to deal with gradual changes in background. PBAS reports a remarkable enhancement of foreground segmentation performance thanks to the use of two controllers with feedback loops for updating the foreground-background decision threshold and the learning rate.

2.2.4 Machine Learning Model

In recent years, the rapid development of machine learning strongly motivates many remarkable types of research in the fields of image processing and computer vision and delivers outstanding outcomes. Obviously, many background modeling and estimation approaches have been proposed with the use of highlight machine learning techniques such as Support Vector Machine, neural

network, and deep learning.

One of the most used machine learning techniques is Support Vector Machine, a supervised learning model for classification and regression analysis. By formulating the problem of background initialization as an online learning problem [65, 66], pixel classification done by a probabilistic SVM. A pixel is examined by SVM to update to the background model, so the background initialization performs until no more detected background pixels. Another machine learning technique received much concentration of researchers for modeling background is the neural network (NN). In [67], a simple perceptron neural network (PNN) is applied to classify background pixel for motion detection. Recurrent neural network (RNN), a probabilistic graphical model, is learned for dynamic background segmentation in [68, 69]. An approach with a self organization through neural networks, namely Self-Organizing Background Subtraction (SOBS) [70–74] is introduced for capturing structural background variation based on periodic-like motion considered in a duration. The model has an ability to handle the dynamic moving background, gradual illumination variation, and stopped foreground in a while. Nowadays deep learning has been widely applied to many kinds of research in image processing and also computer vision. Deep Auto-encoder Networks (DAN) and Convolutional Neural Networks (CNN) are currently two efficient techniques for background modeling. A cascade structure of two deep auto-encoder networks [75, 76], Background Extraction Network (BEN) for initialization and Background Learning Network (BLN) for updating, is proposed for segmenting foreground in a dynamic scene. Several works [77, 78] model the complex and dynamic background image based on learning spatial features by Convolutional Neural Networks (ConvNets).

Chapter 3

Proposed Methodology for Foreground Detection

3.1 Traditional Background Subtraction Methods

3.1.1 Frame Differencing

This is the most simple approach to detect the moving objects in a scene, in which the background image B_t is just estimated to be the previous frame F_{t_1} as follows

$$B_t(x, y) = F_{t-1}(x, y) \tag{3.1}$$

And the foreground is extracted by a general background subtraction equation using the global thresholding as follows

$$|F_t(x, y) - B_t(x, y)| \geq Threshold \tag{3.2}$$

This approach mostly depends on the object structure, speed, frame rate, and the global threshold value. Thus, it is not useful for realistic systems. The general workflow of frame difference approach is shown in Figure 3.1.

3.1.2 Filtering Approach

Another approach which estimates the background reference image is based on the observation of previous N input frames. In particular, the background image is constructed as the average or the median value for each pixel of all three color channels. For example, the background image is

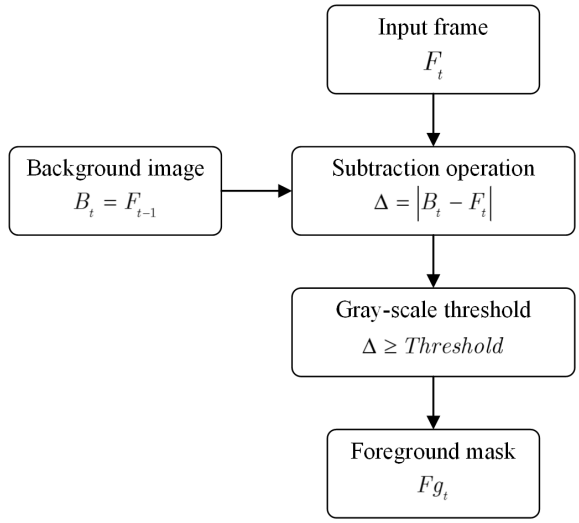


Figure 3.1: The general workflow of frame difference approach.

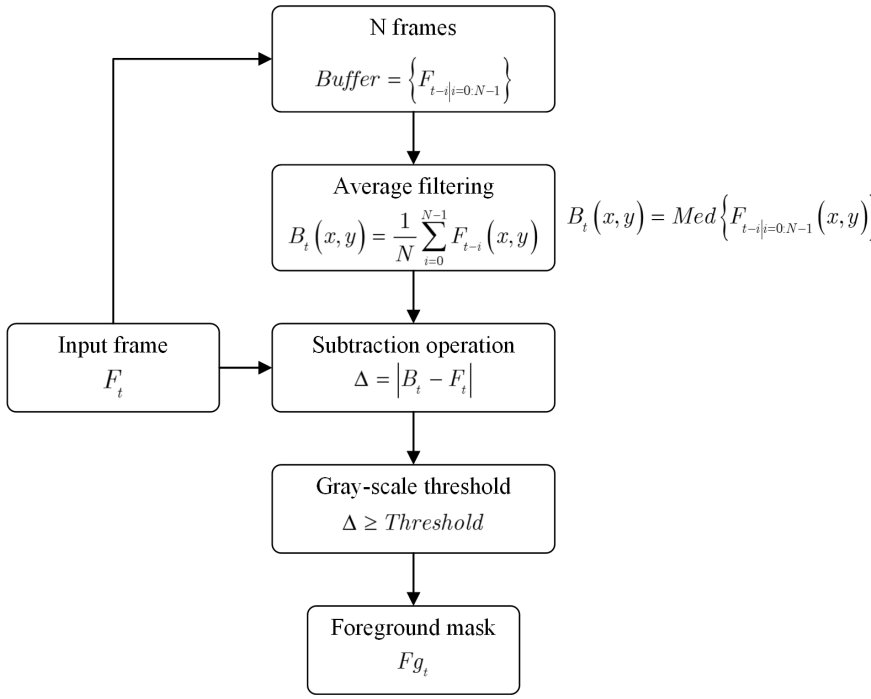


Figure 3.2: A general workflow of average/median filtering approach.

estimated with the average filter as follows

$$B_t(x,y) = \frac{1}{N}\sum_{i=0}^{N-1} F_{t-i}(x,y) \tag{3.3}$$

As another way, the background image is constructed with the median filter as follows

$$B_t(x,y) = Median\left\{F_{t-i|i=0:N-1}(x,y)\right\} \tag{3.4}$$

Median filtering is one of the most commonly-used background modeling techniques. This approach operated with the assumption that the pixels stay in the background for more than half of the frames in the buffer. It can be seen that median filtering can build the background image even the moving objects exist in a scene. In the cases of slow-moving and stopping for a while of objects, the technique requires much more memory for store more frames at a time. With a fixed size of the buffer, the above problem can be partially relieved by collecting input frames at a lower sampling rate. Similar to frame differencing, the foreground is also extracted with the subtraction operation and global thresholding

$$|F_t(x,y) - B_t(x,y)| \geq Threshold \tag{3.5}$$

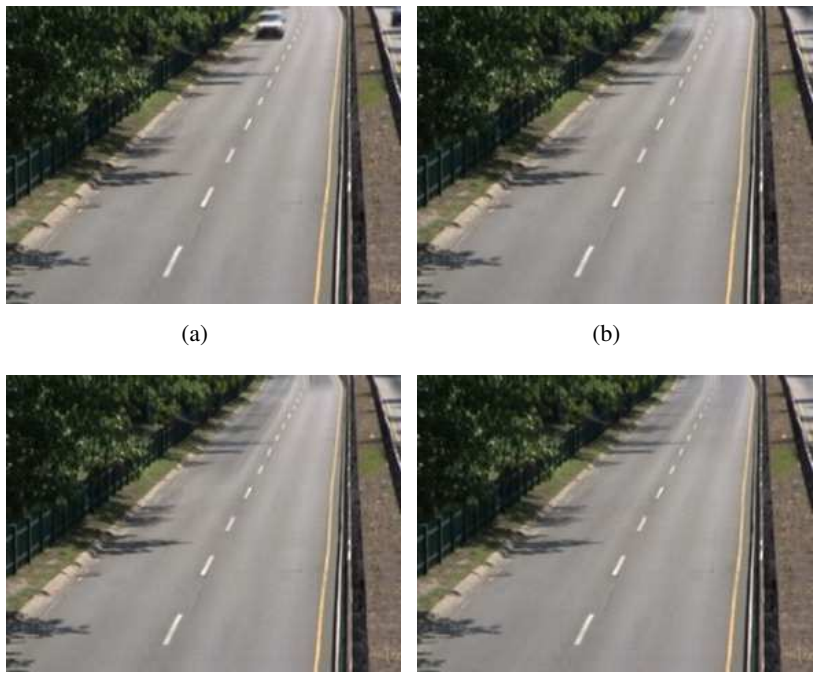
The general workflow of filtering approaches, including average and median, is presented in Figure 3.2 Some examples of background images estimated using average and median filters with various numbers of initial input frames are shown in Figure 3.3 and 3.4, respectively. The quality of background image depends on the size and the sampling rate of the buffer. However, this approach requires the memory to store the previous N input frames and takes time to calculate the average or the median background image at the beginning. In order to handle the issue relating to memory requirement, the running average method is developed, in which the current background reference image is estimated based on the previous background and the current image frame as follows

$$B_t(x,y) = \frac{1}{t}F_t(x,y) + \frac{t-1}{t}B_{t-1}(x,y) \tag{3.6}$$

or more generally

$$B_t(x,y) = \alpha F_t(x,y) + (1-\alpha)B_{t-1}(x,y) \tag{3.7}$$

where α is the learning rate. Fundamentally, the running average based background subtraction



(a) (b)

(c) (d)

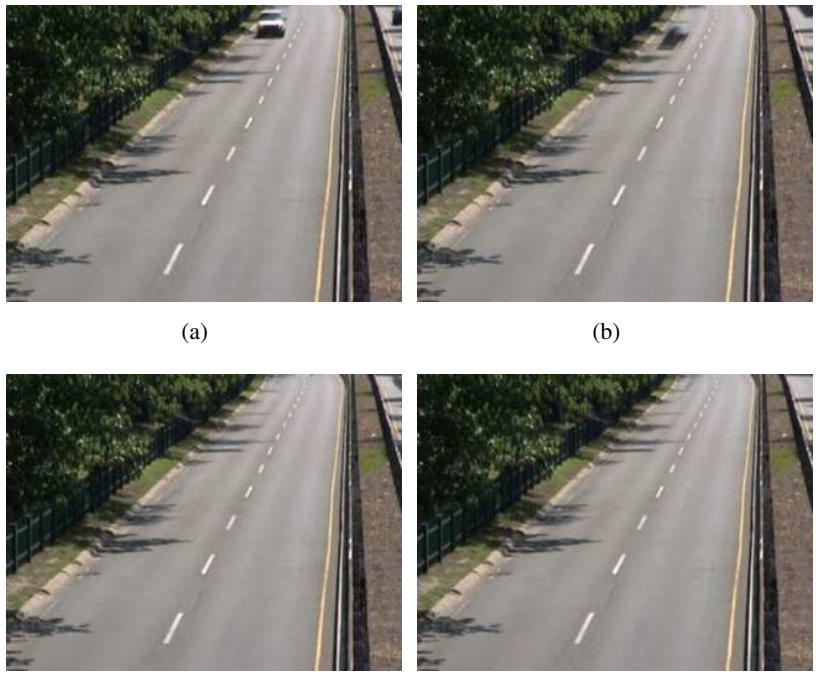
Figure 3.3: The average background images constructed from (a) 10, (b) 50, (c) 100, and (d) 200 input frames.

methods can be performed by following the flowchart in Figure 3.5. To prevent the background model to be polluted by pixel logically not belonging to the background scene, a selective scheme is recommended. For each new frame, the background model is only updated for pixels belonging to background class, that means, updating will be ignored for pixels that are detected as foreground. This selectivity can be expressed as follows

$$B_t(x,y) = \begin{cases} \alpha F_t(x,y) + (1-\alpha) B_{t-1}(x,y) & ; \forall F g_{t-1}(x,y) = 0 \\ B_{t-1}(x,y) & ; \forall F g_{t-1}(x,y) = 1 \end{cases} \tag{3.8}$$

where Fg is the binary foreground image. Similarly, the background image can be approximated as follows

$$B_t(x,y) = \begin{cases} B_{t-1}(x,y) + 1 & ; if \quad F_t(x,y) > B_{t-1}(x,y) \\ B_{t-1}(x,y) - 1 & ; if \quad F_t(x,y) < B_{t-1}(x,y) \end{cases} \tag{3.9}$$



(a) (b)

(c) (d)

Figure 3.4: The median background images constructed from (a) 10, (b) 50, (c) 100, and (d) 200 input frames.

The shortcoming of the updating scheme based on the temporal filters is the appearance of "ghosting" artifacts in the case of slow motion and abandoned object for a while. To solve this problem, Khateeb et al. [79] improve the updating scheme with another parameter to determine how fast or slow the background should be updated as follows

$$B_t(x,y) = \begin{cases} \left\lfloor B^*_{t-1}(x,y) + \frac{1}{\delta} \right\rfloor & ;if \quad F_t(x,y) > B_{t-1}(x,y) \\ \left\lfloor B^*_{t-1}(x,y) - \frac{1}{\delta} \right\rfloor & ;if \quad F_t(x,y) < B_{t-1}(x,y) \end{cases} \tag{3.10}$$

where $\lfloor \rfloor$ is the floor operator, B^* is a real version of the background image, and δ is the updating speed.

Besides the limitation relating to memory consuming, other shortcomings of these simple approaches have been found: (i) only one fixed global threshold is used to classify all pixels in the image, i.e. this value is not a time-wise function, and (ii) the background image is updated with

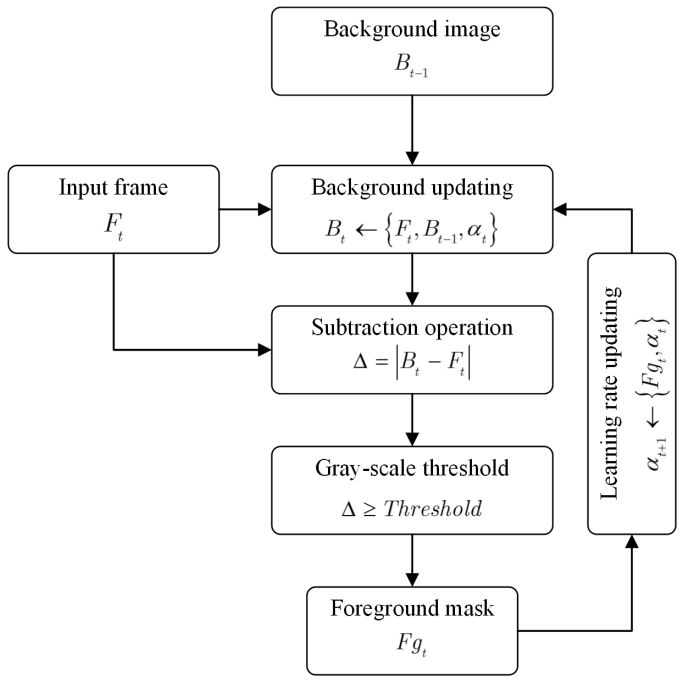


Figure 3.5: The workflow of running average approach.

a predefined learning rate that is also applied to all pixels. Therefore, the threshold value should be selected automatically to adapt to the illumination change. One of the most popular methods to identify the threshold value is Otsu. However, Otsu threshold is slightly sensitive to background noise, especially in the case of non-moving objects in a scene. In order to handle this problem, it is reasonable to set the lower and upper thresholds for this parameter. Moreover, an adaptive learning rate is considered to overcome the illumination change in a scene. For example, Kang et al. [17] updates α based on the threshold value and foreground detection result as

$$\alpha = \frac{Threshold}{2.5 \times 255 \times m} \tag{3.11}$$

The value of m stands for the moving speed and identified as following to overcome the problem of background-object mixture.

$$m = \frac{S_n - S_a}{S_n} \tag{3.12}$$

where

$$\begin{aligned} S_a &= \sum_{x=1}^{P} \sum_{y=1}^{Q} Fg_{\Delta}(x,y) \\ S_n &= \sum_{x=1}^{P} \sum_{y=1}^{Q} Fg_t(x,y) \\ Fg_{\Delta}(x,y) &= Fg_t(x,y) \cdot Fg_{t-1}(x,y) \end{aligned} \tag{3.13}$$

where F is the foreground mask, represented in binary, and $(\cdot)$ represents for the AND operator. To avoid that the background is reconstructed too fast or too slowly, the limit of m should be established as follows

$$m = \begin{cases} \max & ;if \quad m \geq \max \\ \min & ;if \quad m \leq \min \end{cases} \tag{3.14}$$

3.1.3 Running Gaussian Average Approach

As a simple manner of background representation, running Gaussian average approach models the background as a Gaussian distribution of the history over time of pixel intensity [29]. The background is separately modeled for each color channel with the rationale of fitting a Gaussian probability density function (pdf) on the last N values of each pixel. As the results, there are two images which hold the mean and the standard deviation for each channel. In particular, the background model is initialized after gathering N input frames and only updated through two parameters instead of fitting the pdf from scratch at each new frame for speed and accuracy increment. The mean and standard deviation are initially estimated for each pixel from N input frames as follows

$$\begin{aligned} \mu(x,y) &= \tfrac{1}{N} \sum_{i=0}^{N-1} F_t(x,y) \\ \sigma^2(x,y) &= \tfrac{1}{N} \sum_{t=1}^{N} \left(F_t(x,y) - \mu(x,y)\right)^2 \end{aligned} \tag{3.15}$$

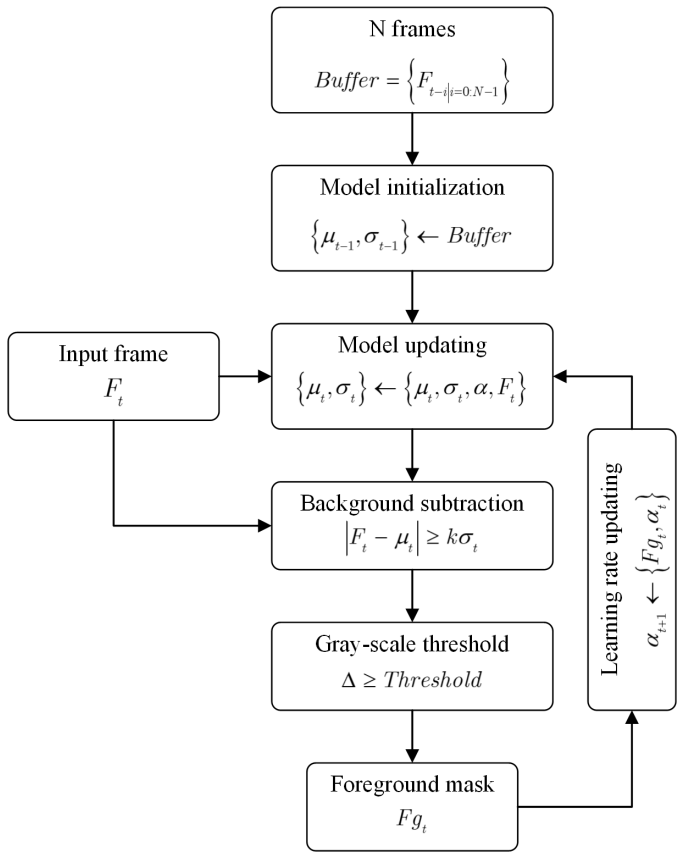


Figure 3.6: The general workflow of running Gaussian average.

For each input frame, the Gaussian background model is updated by running average:

$$\begin{aligned} \mu_t(x,y) &= \alpha F_t(x,y) + (1-\alpha)\,\mu_{t-1}(x,y) \\ \sigma_t^2(x,y) &= \alpha(F_t(x,y) - \mu_{t-1}(x,y))^2 + (1-\alpha)\,\sigma_{t-1}^2(x,y) \end{aligned} \tag{3.16}$$

At each input frame, a pixel can be then classified to the foreground class by the following inequality

$$|F_t(x,y) - \mu_t(x,y)| \geq k\sigma_t(x,y) \tag{3.17}$$

otherwise, a pixel will be classified to the background class. In [80], Koller et al. improved the model updating scheme based on the foreground mask detected in the previous frame $t-1$.

$$\mu_t = Fg_{t-1}\mu_{t-1} + (1 - Fg_{t-1})(\alpha F_t + (1-\alpha)\mu_{t-1}) \tag{3.18}$$

It can be seen that this updating scheme is similar to the selective updating presented in (2.8). For instance, if a pixel belongs to background class, i.e. $Fg_{t-1}(x,y) = 0$, the updating is similar to (2.16). In contrast, if a pixel is detected as a foreground pixel, i.e. $Fg_{t-1}(x,y) = 1$, the background model is maintained without updating. Typically, the learning rate α can be fixed as a constant value which is usually less than the sample rate. However, the smaller the updating rate, the slower a foreground detection system respond to the actual background dynamic. Some researchers recommend an adaptive learning rate. In [30], the learning rate is defined based on the threshold of foreground detection as follows

$$\alpha_{t+1} = \begin{cases} \alpha_t - \frac{Threshold_t \times \alpha_t}{\varsigma} & ;if \quad Threshold_t \leq Threshold_{t-1} + \tau \\ \alpha_t + \frac{Threshold_t \times \alpha_t}{\varsigma} & ;if \quad Threshold_t \geq Threshold_{t-1} - \tau \end{cases} \tag{3.19}$$

where $Threshold$ is determined by Otsu method, τ represent tolerance in threshold variation, and ς is the correction rate. The value of τ defines how much the variation of threshold value allowed for adjusting α while ς controls the adaptation of α. The general workflow of running Gaussian average is seen in Figure 3.6.

3.1.4 Histogram Over Time

Another approach considered to estimate the background reference image is histogram over time. After accumulating N input frames to a buffer, the histograms of each pixel are analyzed for all three color channel. As an assumption, the histogram of a static pixel that belongs to the background class is distributed in a small variance due to illumination changes. These static background pixels are perceived as outliers and therefore eliminated from the processing. The intensity value with the highest frequency will be defined as the background intensity value. For example, the intensity of 6 getting the most probability over 50 frames in Figure 3.7 is defined as the inten-

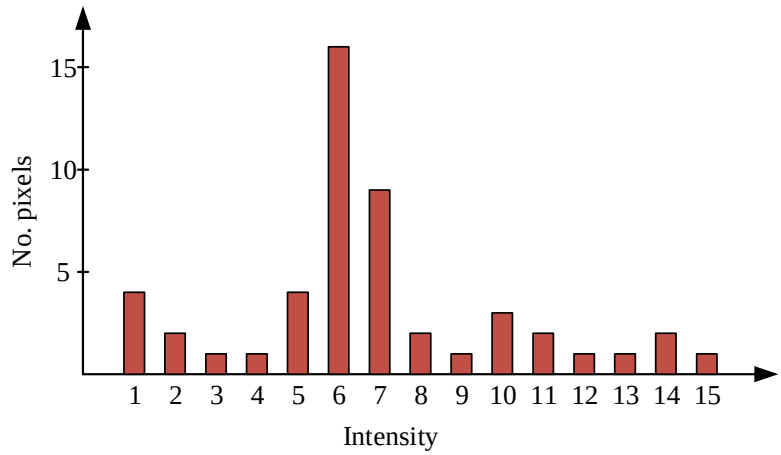


Figure 3.7: An example of histogram over time representation of a pixel.

sity of the corresponding pixel in the background image. Generally, the background image can be formulated as follows

$$\begin{aligned} B(x,y) &= n \,|pdf_n(x,y) = \max\{pdf_{i=0:L}(x,y)\} \\ pdf_i(x,y) &= \tfrac{n_i(x,y)}{N} \end{aligned} \tag{3.20}$$

where g is the intensity at which its probability density function is maximum, L is the number of intensity levels $(= 255)$, and n_i is the number of pixels at coordinate (x, y) having the intensity i. It should be noted that $\sum_{i=0}^{L} pdf_i(x,y) = 1$. The histogram over time of each pixel is updated for each input frame coming as follows

$$H_{t+1}(x,y) = \begin{cases} update\left(H_t(x,y)\right) & ;if \quad Fg_t(x,y) = 0 \\ H_t(x,y) & ;if \quad Fg_t(x,y) = 1 \end{cases} \tag{3.21}$$

Compared to other background subtraction methods in the category of basic model, histogram over time requires more memory for storing the histogram information, particularly depends on the image resolution, and also expensive computation for updating histogram of all pixels in three color channels.

3.2 Overview of Proposed Approach

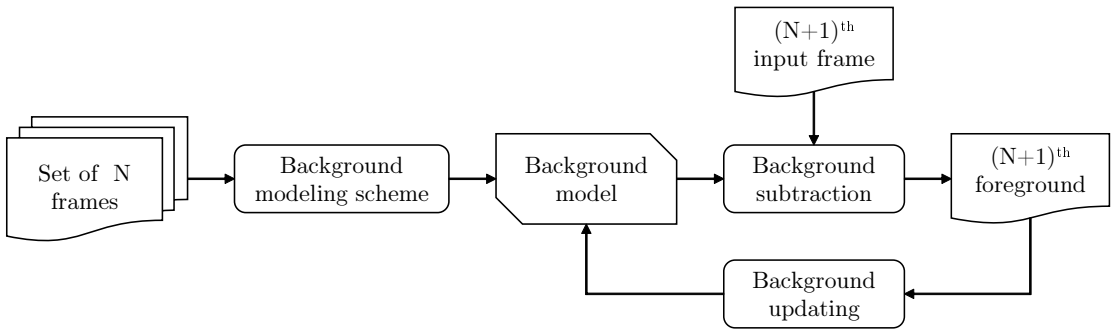


Figure 3.8: The general framework of other existing background subtraction approaches with the main contribution of background modeling scheme.

Generally, the efficient method which is used for detecting foreground based on the background subtraction scheme includes two main phases: i) the background estimation and ii) the foreground extraction. In the thesis, we contribute on the background estimation phase with a novel algorithm, namely Neighbor-based Intensity Correction (NIC), that has the ability to estimate the background image from the previous background and the current input frame. Most of the existing approaches in the field of background subtraction mainly contribute the background modeling algorithm which builds an initial background model based on a deterministic amount of input frames as an illustration in Figure 3.8. Those models are updated based on the extracted foreground using an updating scheme which is specified by a learning rate. It is realized that those approaches are memory- and time-consuming for constructing the background model. Furthermore, they can only extract the foreground after a pre-determined set of initial input frames instead of at the beginning of a video. Different from them, our proposed NIC algorithm assumes the first frame of a video as the initial background image and then updates it directly in time for each input frame as shown in Figure 3.9. The idea of NIC is that the algorithm replaces the object intensity by the background intensity at detected motion pixels. By this way, the objects appearing in the background image are eliminated out of the scene in time, hence the foreground is extracted more precisely. Due to directly updating the background image instead of the background model likes others, NIC uses less memory. Additionally, NIC allows extracting the foreground at the second input frame henceforth. After updating the background image with NIC algorithm, the foreground is detected

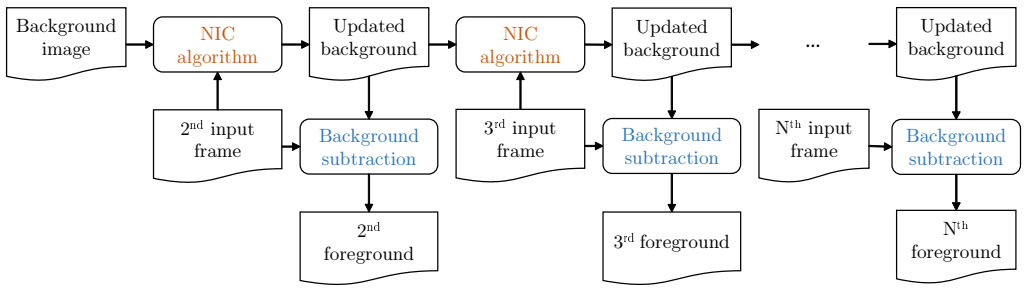


Figure 3.9: The framework of our proposed background subtraction method with the contribution of NIC algorithm as a background estimation and updating scheme.

by subtraction operation between the input frame and the updated background. The foreground is then segmented by Otsu thresholding method and finalized with some morphological operations as the post-processing.

3.3 Background Estimation Algorithm

The algorithm is proposed with two assumptions to ensure its consistency and correct operation: (i) the algorithm updates the background image successfully whenever object movement is detected in a scene and (ii) the background is more homogeneous in the intensity than the object in local. Generally, NIC algorithm consecutively models and updates the background image from the motion information extracted from the difference between the input frame and the background image. The workflow of the foreground detection using our proposed NIC algorithm for background estimation is concretely represented in Fig. 3.10 with the frame sequence as the input and the updated background image as the output.

3.3.1 Background Initialization

As the prior knowledge, the first frame of an image sequence is assumed as the initial background reference image:

$$B_1(x, y) = F_1(x, y) \tag{3.22}$$

where $B_1(x, y)$ and $F_1(x, y)$ are the intensity values of the pixel that is located at the coordinate (x, y) with $x \leq P$ and $y \leq Q$ in the initial background image and the first frame, where P and Q are the width and height of the input frame, respectively. With this assumption, only the background reference image needs to be stored in the memory, thus, it is so efficient of memory in use if compared with other basic models. It should be noted that this initial background image can contain the moving objects in a scene or not. Therefore, the role of NIC algorithm is to eliminate the moving objects out of the background image by replacing the object intensity by the background intensity at corresponding motion pixels. This process can be called as the background updating which is performed consecutively for each input frame.

3.3.2 Background Updating

For each input i^{th} frame, where $i \geq 2$, the background image B_i is estimated from B_{i-1} and the input frame F_i by our proposed NIC algorithm In the first step of NIC algorithm, the difference between the current background image B_{i-1} and the input frame F_i, denoted D_i, is calculated by the following equation for the grayscale image:

$$D_i(x, y) = |F_i(x, y) - B_{i-1}(x, y)| \quad ; \forall i \geq 2 \tag{3.23}$$

It is recognized that the difference image D_i contains information about moving objects and noise, especially in the dynamic background videos such as water wave, leaf movement, and etc.; hence D_i should be clustered into the motion and non-motion regions by a constant threshold τ:

$$D_i^*(x, y) = \begin{cases} 1 & ; \forall D_i(x, y) \geq \tau \\ 0 & ; \forall D_i(x, y) < \tau \end{cases} \tag{3.24}$$

This thresholding step aims to identify the preliminary motion pixels which need to be updated in the intensity by a proposed intensity updating scheme. Figure 3.11 shows an example of difference image D_i in gray-scale and D_i^* in binary. The binary image D_i^* has 0-bit pixels representing the non-motion areas and 1-bit pixels representing the motion areas. In principle, moving objects have a greater difference than noise and shadow artifacts. A higher value of the constant threshold

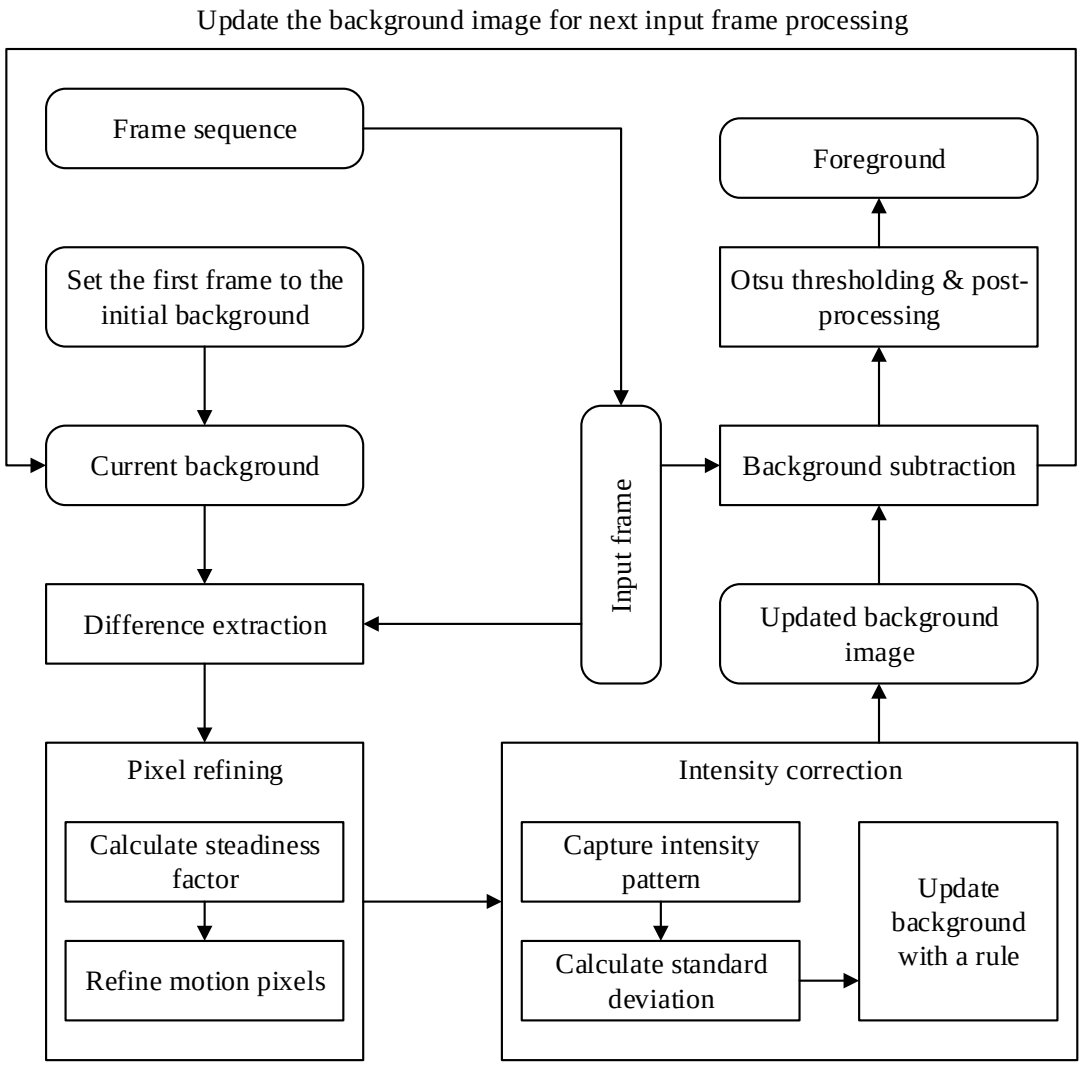
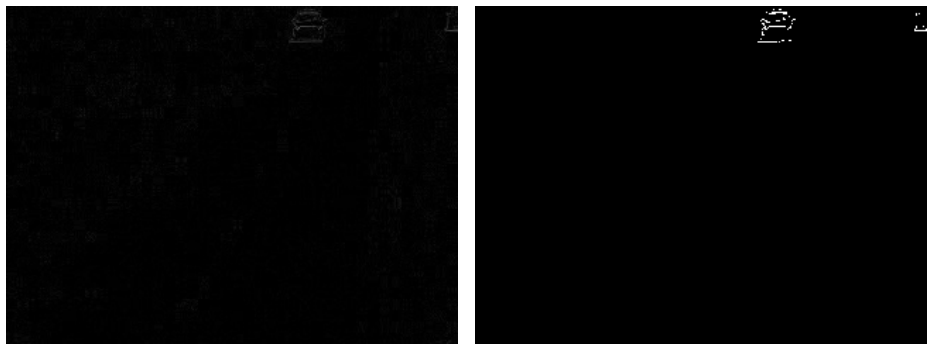


Figure 3.10: The workflow of background subtraction method, wherein the background image is estimated by the NIC algorithm.

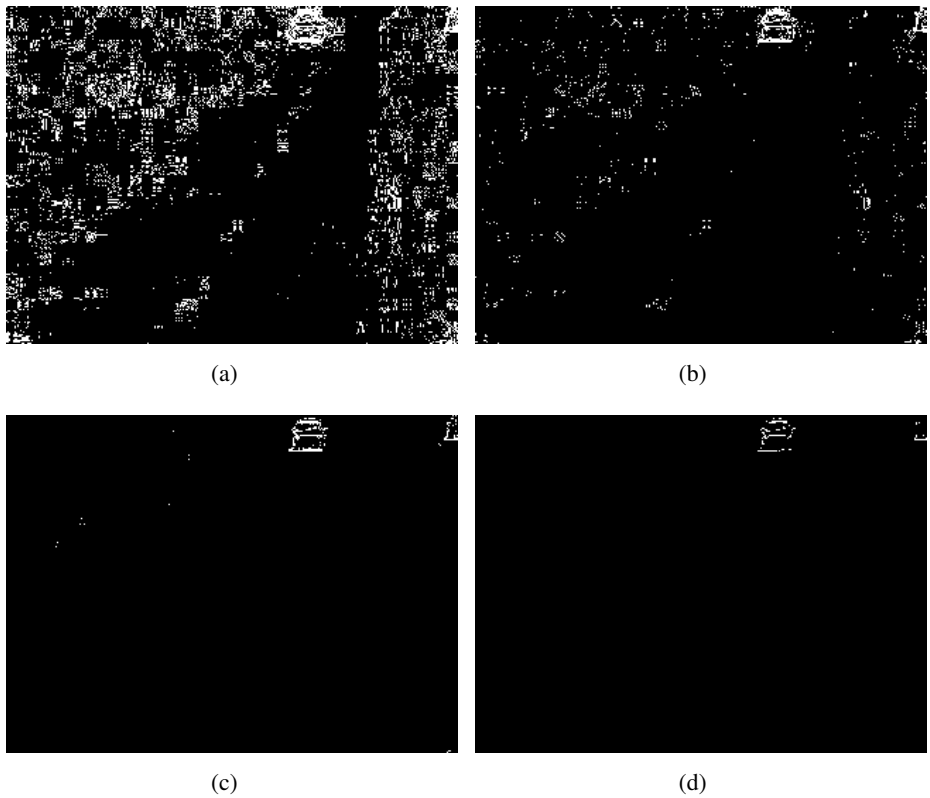


(a) (b)

Figure 3.11: The examples of (a) difference image D_i and (b) $\mathcal{D}_i$ after thresholding with $\tau = 30$

τ eliminates the noise, but nevertheless, some motion pixels cannot be detected undesirably. In contrast, if τ is low, noise pixels are sometimes misclassified as motion pixels. In Figure 3.12, we present some examples of different image D_i^* which are thresholded with different values of τ. It can be observed that with $\tau = 5$ as Figure 3.12(a) there are so many noise pixels appearing in the background identified as the motion pixels. Updating the intensity of these noise pixels is meaningless with expensively computational consumption and harms to the precision of updating truthful motion pixels potentially. By increasing the value of τ, we can eliminate noise and only keep the object motion pixels as illustrated in Figure 3.12(b), (c), and (d). However, some useful information of moving objects in a scene can be ignored without intensity update. For instance, the edge information of an object in Figure 3.12 is missing with $\tau = 30$. Therefore, the threshold τ, which obviously has an effectiveness on the result of D_i^*, should be carefully selected through experimental evaluations.

Principally, the proposed intensity updating scheme will be applied for motion pixels in a set of $D_i^* = 1$; however, to reduce the appearance of outliers as much as possible, we consider a novel factor, called steadiness factor, which is able to measure the frequency of change of pixel intensity between the input frame F_i and the background image B_{i-1}. The consistence of the background image is also examined by this factor. The steadiness factor of a pixel (x, y), denoted $S(x, y)$, is updated consecutively for each input frame. If there exist intensity changes in the input frame comparing to the background image, the corresponding pixel is less steady than that remains the



(a) (b)

(c) (d)

Figure 3.12: An example of difference image D_i is segmented with various threshold values: (a) $\tau = 5$, (b) $\tau = 10$, (c) $\tau = 20$, (d) $\tau = 30$

same. Concretely, the steadiness value of a pixel is accumulated in an image sequence as follows:

$$S_i(x,y) = \begin{cases} S_{i-1}(x,y) - 1 & ;\forall D_i^*(x,y) = 1 \\ S_{i-1}(x,y) + 1 & ;\forall D_i^*(x,y) = 0 \end{cases} \tag{3.25}$$

where S_i is the steadiness matrix at the i^{th} frame. It is initialized with a value of zero, i.e., $S_1(x,y) = 0$ for the all pixels. It should be noted that the size of steadiness matrix equals to the input frame resolution. In the accumulation of frame by frame, the steadiness value of a non-motion pixel is greater than those of a motion pixel. For example, an arbitrary pixel p is detected as the motion pixel for t_1 frames, i.e. $D^*_{[t_0,t_0+t1-1]}(x_p,y_p) = 1$ and as the non-motion pixel for t_2 frames, i.e. $D^*_{[t_0+t_1,t_0+T-1]}(x_p,y_p) = 0$ after $T = t_1 + t_2$ frames. Calculated by (3.25), the steadiness value $S_T(x_p,y_p) = (-1)\,t_1 + (1)\,t_2$ is negative if $t_1 > t_2$, positive if $t_1 < t_2$, and zero

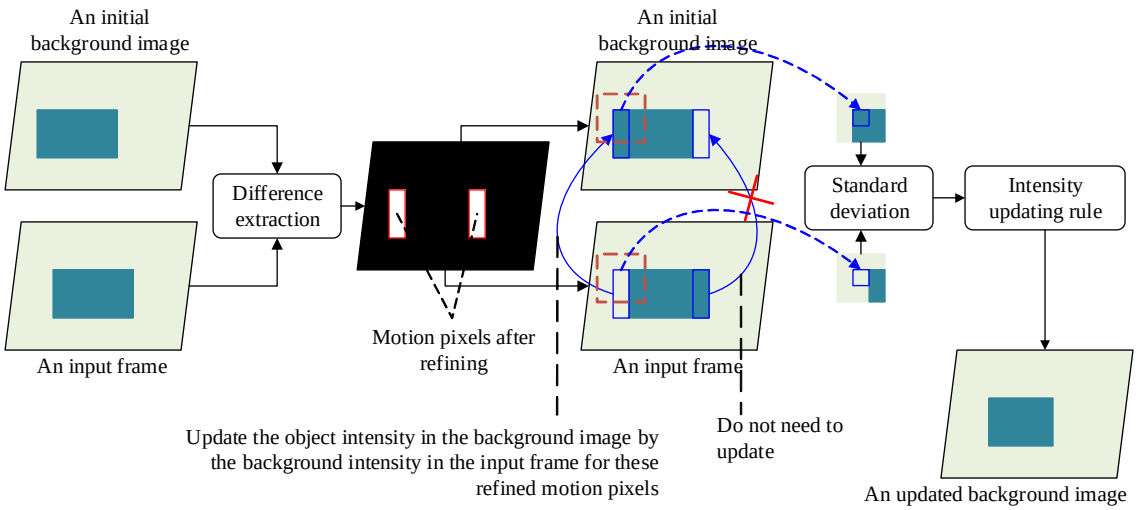


Figure 3.13: The idea of NIC algorithm for pixel intensity updating.

if $t_1 = t_2$.

Based on the steadiness matrix S, the NIC algorithm can measure the consistence of the current background image. During the estimation process, the intensity updating process might not be necessary for all detected motion pixels. In particular, we focus on updating the intensity for only motion pixels having negative steadiness value. For example with the non-object background image, if irregular motion such as water wave, leaf movement, and etc. is detected by the first step using (3.23) and (3.24), the motion pixels can be identified with the positive steadiness values. Updating intensity for these pixels is not necessary and takes more computational resource. By crossing two conditions of motion pixels and negative steadiness values, we define a set of pixels for applying NIC algorithm:

$$P_i = \{(x, y) \,|[D_i^* (x, y) = 1 \cap S_i (x, y) < 0]\} \tag{3.26}$$

where P is a set of refined motion pixels. Another benefit of this pixel refining step is to reduce the computational cost of NIC algorithm for motion pixels generated from the sudden light changes or artifacts.

The intensity correction algorithm is executed for the pixels in the refined set P_i. The main idea of the proposed updating scheme is illustrated in Figure 3.13. As an assumption, since the

initial background image contains moving objects or not, the idea of NIC algorithm is to eliminate these moving objects out of the background image by replacing the object intensity by the background intensity at corresponding pixels. In the example in Figure 3.13, it can be observed that three refined motion pixels on the left side should be updated by the background intensity in the input frame for the background image at corresponding locations while three refined motion pixels on the right side do not need to be modified. However, the updating process is performed with unknown information of the role of these motion pixels, that means, we do not know the value of motion pixel in the background image represents the background intensity or the object intensity. Making update decision is therefore done for a motion pixel by examining the difference between two intensity patterns of surrounding neighbor pixels, that are extracted from both the background image and the input frame by a square mask. Generally, the background is more homogeneous in the intensity than the foreground or the object region. Based on comparing the homogeneity of two patterns together, we can identify the intensity of the motion pixel should be updated from either the background pattern or the input frame pattern. In this research, the standard deviation is utilized to measure the homogeneity of intensity values in a pattern. Fundamentally, a smaller standard deviation represents a greater homogeneity of a set of values. At the next step, the standard deviation metric of intensity values is calculated for two corresponding patterns. By comparing the standard deviation, the intensity update is done by a proposed updating rule.

Following the workflow of NIC algorithm presented in Figure 3.10, for each refined motion pixel in set P, two intensity patterns are captured from the background image and the input frame by a square mask of size $(n \times n)$, denote $W^B_{(x,y)}$ and $W^F_{(x,y)}$, respectively. To differentiate two intensity patterns, we compare the homogeneity of intensity based on the standard deviation metric. At the next step, two values of standard deviation of intensity values are calculated from two patterns, respectively. Fundamentally, the standard deviation σ is defined as follows:

$$\sigma = \sqrt{\frac{1}{N}\sum_{p=1}^{n}\sum_{q=1}^{n}\left(I\left(p,q\right)-\mu\right)^2} \tag{3.27}$$

where n is the size of a square mask and $N = n^2$ is the total number of pixels. The mean value μ

of pixels within a mask is determined as follows:

$$\mu = \frac{1}{N} \sum_{p=1}^{n} \sum_{q=1}^{n} I\left(p, q\right) \tag{3.28}$$

For each refined motion pixel (x, y) in set P_i, two standard deviation values, denoted $\sigma^B_{(x,y)}$) and $\sigma^F_{(x,y)}$, are calculated from the two patterns of the current background $W^B_{(x,y)}$ and the current frame $W^F_{(x,y)}$. The proposed intensity updating rule is based on the result of the comparison process between the two standard deviation values. Concretely, the motion pixel will be updated by the value whose intensity pattern is more homogeneous, that means, whose the standard deviation is smaller because. The updating rule is described as follows:

$$B_i\left(x, y\right) = \begin{cases} B_{i-1}\left(x, y\right) & \text{if} \quad (x, y) \notin P_i \\ B_{i-1}\left(x, y\right) & \text{if} \quad (x, y) \in P_i \left| \sigma^F_{(x,y)} \geq \sigma^B_{(x,y)} \right. \\ F_i\left(x, y\right) & \text{if} \quad (x, y) \in P_i \left| \sigma^F_{(x,y)} < \sigma^B_{(x,y)} \right. \end{cases} \tag{3.29}$$

According to (3.29), if the background pattern $W^B_{(x,y)}$ is more homogeneous than the input frame pattern $W^F_{(x,y)}$, i.e. $\sigma^B_{(x,y)} \leq \sigma^F_{(x,y)}$, the pixel at the corresponding location in the background image $B_{i-1}\left(x, y\right)$ currently represents background intensity. No intensity modification is done for the motion pixel in the updated background image. In contrast, if $\sigma^F_{(x,y)} < \sigma^B_{(x,y)}$, i.e. the input frame pattern is more homogeneous than the background pattern, we know that the currently examining pixel in the background image $B_{i-1}\left(x, y\right)$ needs to be updated by the intensity of the pixel at the corresponding location in the input frame $F_i\left(x, y\right)$.

In order to much more understand the operation of NIC algorithm, a simple example is clarified in Figure 3.14. The initial background image is assumed as the first frame of a video. Therefore NIC algorithm is able to consider the second input frame for extracting the foreground. In this example, the background class has the intensity g_0 and the object class has the intensity g_1. The object appearing in a scene is moving from left to right side. After refining motion pixels by (3.26), there are six pixels which are identified for updating intensity. It can be seen that after successfully updating for the first pixel the updated background image is then used for next turn with the second motion pixels. After updating accomplishment, three object pixels on the left side are modified by

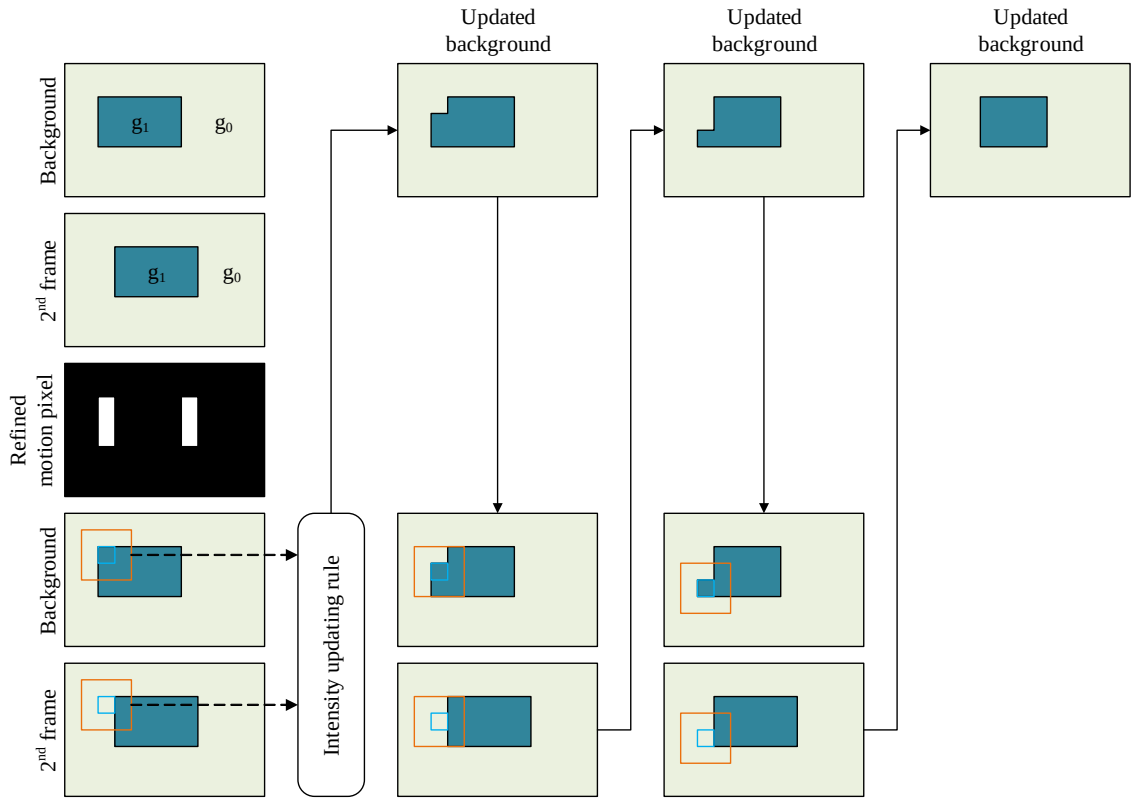


Figure 3.14: An example of intensity updating by NIC algorithm using the (3×3), in which the background image is assumed as the first frame of an input video. NIC algorithm starts to update this background image from the 2^{nd} frame.

the background intensity from the input frame.

In the pattern capturing, the size of mask $(n \times n)$ can be adjusted for adapting to various different situations. To investigate the effectiveness of mask size on the result of the intensity correction rule, a particular example is represented. Let n_0 and n_1 be the number of non-motion as background and motion as object pixels in a pattern with the intensity g_0 and g_1, respectively. So the pattern contains $(n_0 + n_1)$ pixels. The mean of the intensity values is calculated by (3.28) is:

$$\mu = \frac{g_0 n_0 + g_1 n_1}{n_0 + n_1} \tag{3.30}$$

The standard deviation is also defined by another form of (3.27):

$$
\begin{aligned}
\sigma &= \sqrt{\frac{1}{N}\sum_{j=1}^{N}(I_j-\mu)^2} \\
&= \sqrt{\frac{1}{n_0+n_1}\left[n_0(g_0-\mu)^2+n_1(g_1-\mu)^2\right]} \\
&= \sqrt{\frac{1}{n_0+n_1}\left[n_0\left(g_0-\frac{n_0g_0+n_1g_1}{n_0+n_1}\right)^2+n_1\left(g_1-\frac{n_0g_0+n_1g_1}{n_0+n_1}\right)^2\right]} \\
&= \sqrt{\frac{1}{n_0+n_1}\left[n_0\left(\frac{n_1\left(g_0-g_1\right)}{n_0+n_1}\right)^2+n_1\left(\frac{n_0(g_1-g_0)_1}{n_0+n_1}\right)^2\right]} \\
&= \sqrt{\frac{n_0n_1\left(n_0+n_1\right)\left(g_0-g_1\right)^2}{\left(n_0+n_1\right)^3}} \\
&= \frac{\sqrt{n_0n_1}}{(n_0+n_1)}\left|g_0-g_1\right|
\end{aligned}
\tag{3.31}
$$

Using (3.31) for the standard deviation calculation, the term $|g_0 - g_1|$ and $(n_0 + n_1)$ are constant components, so the decision is made by the term $\sqrt{n_0 n_1}$, i.e. the standard deviation value depends on the number of motion and non-motion pixels. In Figure 3.14, the background is updated with the mask size of (3×3). For the first motion pixel updating, $W^B_{(x,y)}$ captured from the background image contains five non-motion pixels and four motion pixels while $W^F_{(x,y)}$ covers seven non-motion pixels and two motion pixels captured from the input frame. Because $\sigma^F_{(x,y)} < \sigma^B_{(x,y)}$, the pixel intensity is modified from g_1 to g_0 at the background image by referring to (3.29). However, NIC sometimes fails to update intensity in the cases of the object area is more homogeneous than the background area in local and the mask size is not large enough to only cover the object area. That leads to an erroneous background updating where a part of object area is now perceived as background. The failure of updating is illustrated as an example in Figure 3.15. In particular, for the second motion pixel updating with 3^{rd} frame, the pixel intensity is modified from g_0 to g_1 at the background image because all pixels in $W^F_{(x,y)}$ present the object intensity g_1, that means, the pattern $W^F_{(x,y)}$ is more homogeneous than the pattern $W^B_{(x,y)}$. A similar incident occurs in the updating process for 4^{th} frame.

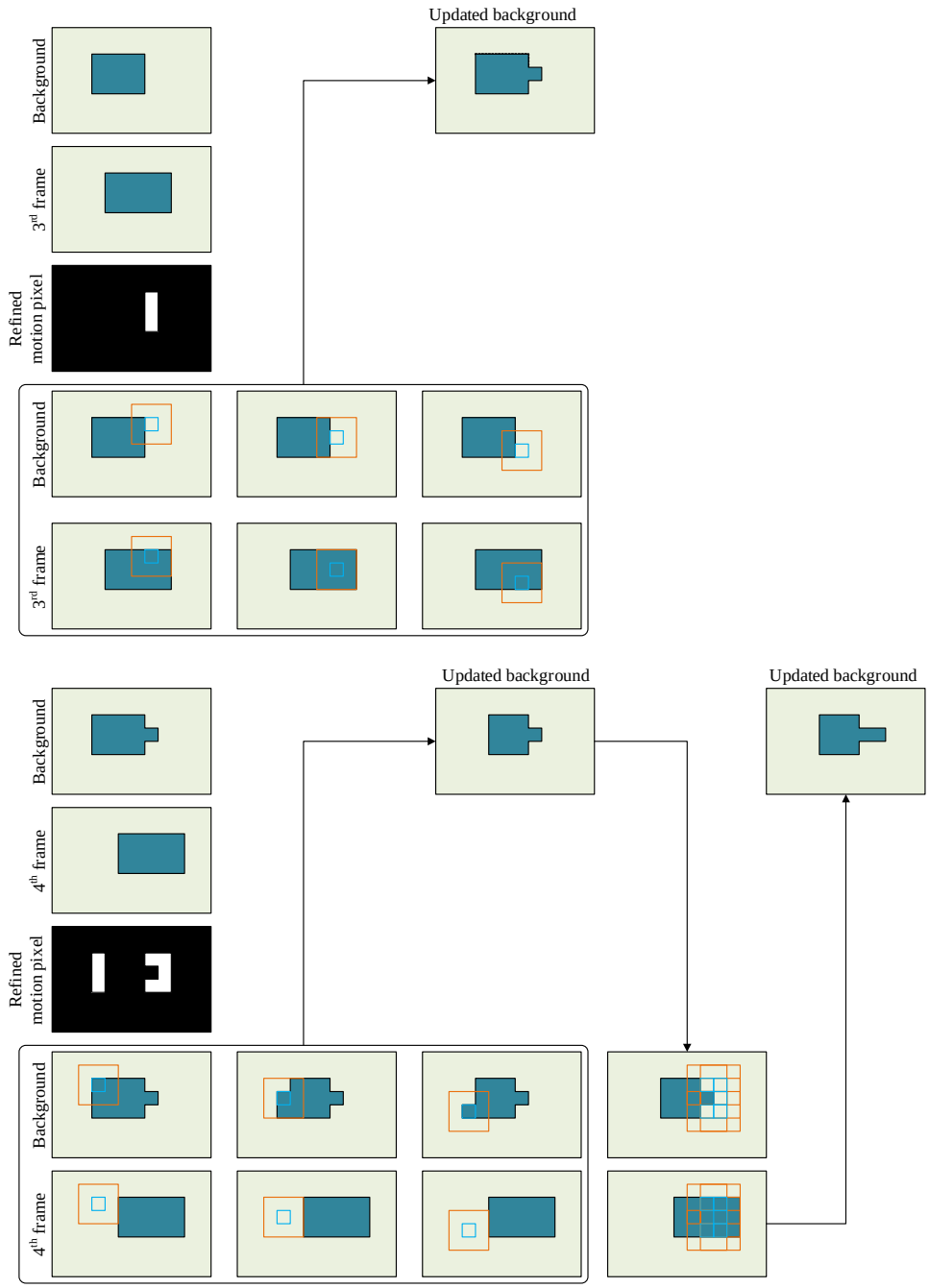


Figure 3.15: Continue the example in Figure 3.14, in which NIC fails to update the background intensity for 3^{rd} (top) and 4^{th} (bottom) frame due to a more homogeneity of the object intensity from the input pattern.

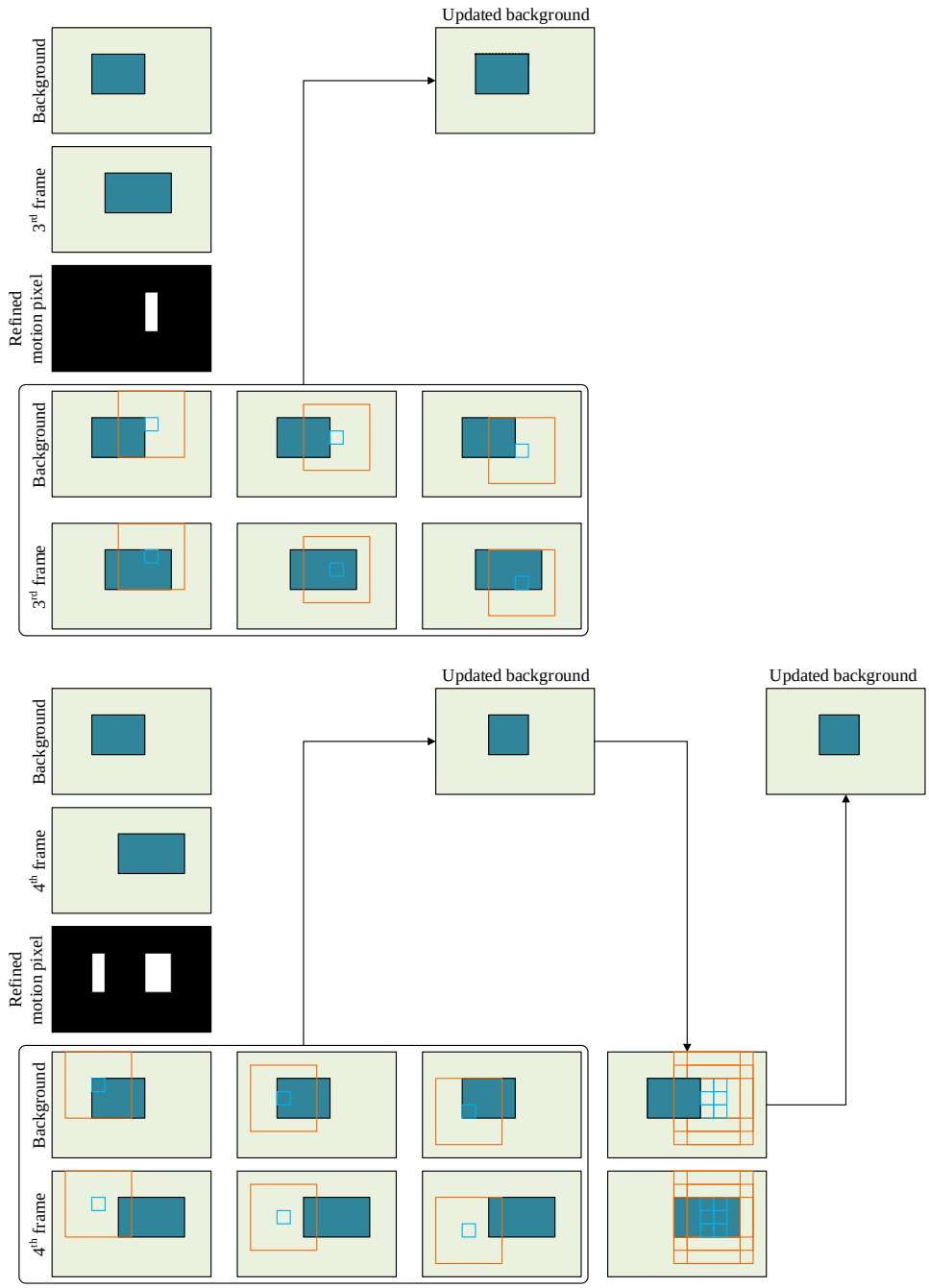


Figure 3.16: An example of intensity updating by NIC algorithm using the mask (5×5) to overcome the failure when using mask (3×3).

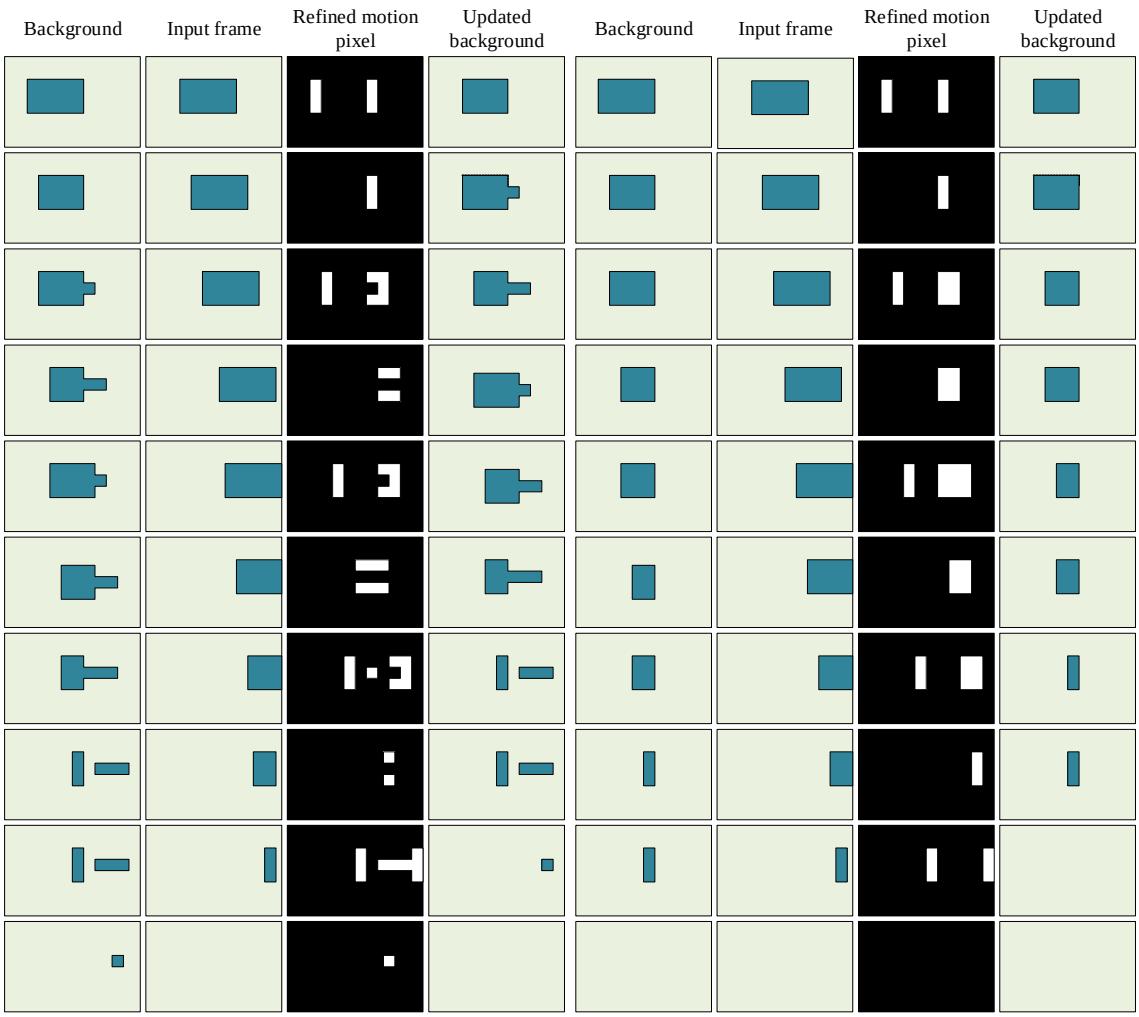


Figure 3.17: Comparison of background updating result using the mask (3×3) (left) and the mask (5×5) (right). Compared to (3×3), (5×5) updates the background image more accurate and faster to approach the non-object background. For instance, NIC algorithm entirely eliminates objects out of the background image at 11^{th} frame if using mask (3×3), meanwhile only nine times of intensity updating are needed for objects elimination if using mask (5×5).

The drawback discussed above is able to be solved by adjusting the mask size which is for capturing the intensity pattern surrounding motion pixel. In the examples illustrated in Figure 3.16, we expand the size from (3×3) to (5×5). This adjustment is useful for NIC algorithm to update intensity correctly for motion pixels as shown in Figure 3.16 for 3^{rd} (top) and 4^{rd} (bottom) input frame. For instance with the second motion pixel updating in 3^{rd} frame, the ratios of motion to non-motion pixels in two patterns are changed significantly if compared to the example using mask (3×3) in Figure 3.15. Clearly, the quality of the background images after updating intensity with NIC algorithm at 3^{rd} and 4^{th} frames using mask (5×5) frame is better than the results achieved by mask (3×3). Besides the benefit of higher estimation accuracy, using mask (5×5) further helps to achieve a stable background image faster if compared to using mask (3×3). We further present the background updating results of a short sequence with 11 frames (including the first frame assumed as the initial background) using the mask (3×3) in Figure 3.17 (left) and the mask (5×5) in Figure 3.17 (right). One important thing should be noticed that failed updating pixels are still considered for updating again in several next frames because they are realized as motion pixels, if compared with the input frames, with negative steadiness values. Although (5×5) is better than (3×3) in estimation accuracy as above analysis, it takes more processing time for calculating standard deviation in updating process. This is the trade-off between accuracy and computational complexity of NIC algorithm.

Using a larger size of a mask for extracting intensity pattern is further useful for the case of multipixel shifting motion that usually occurs in the practical environment. For instance, an object comes in a scene and moves with different velocities as an illustration shown in Figure 3.18. It can be seen that several motion pixels are erroneously updated in the background image, highlighted by red bounding box in Figure 3.18 (left), i.e. from g_0 to g_1, in the case of (3×3). That degrades the quality of updated background image and postpones the acquirement of a non-object background. Meanwhile, with the mask (5×5) as shown in Figure 3.18 (right), the proposed algorithm updates the background image more accurate. Based on above detail analysis, it is realized that the mask size is an important impact of NIC algorithm because making the decision for updating is dependent on standard deviation σ. Therefore, this parameter needs to be reasonably selected to be adaptive to various objects which moves with different speeds. Some experimen-

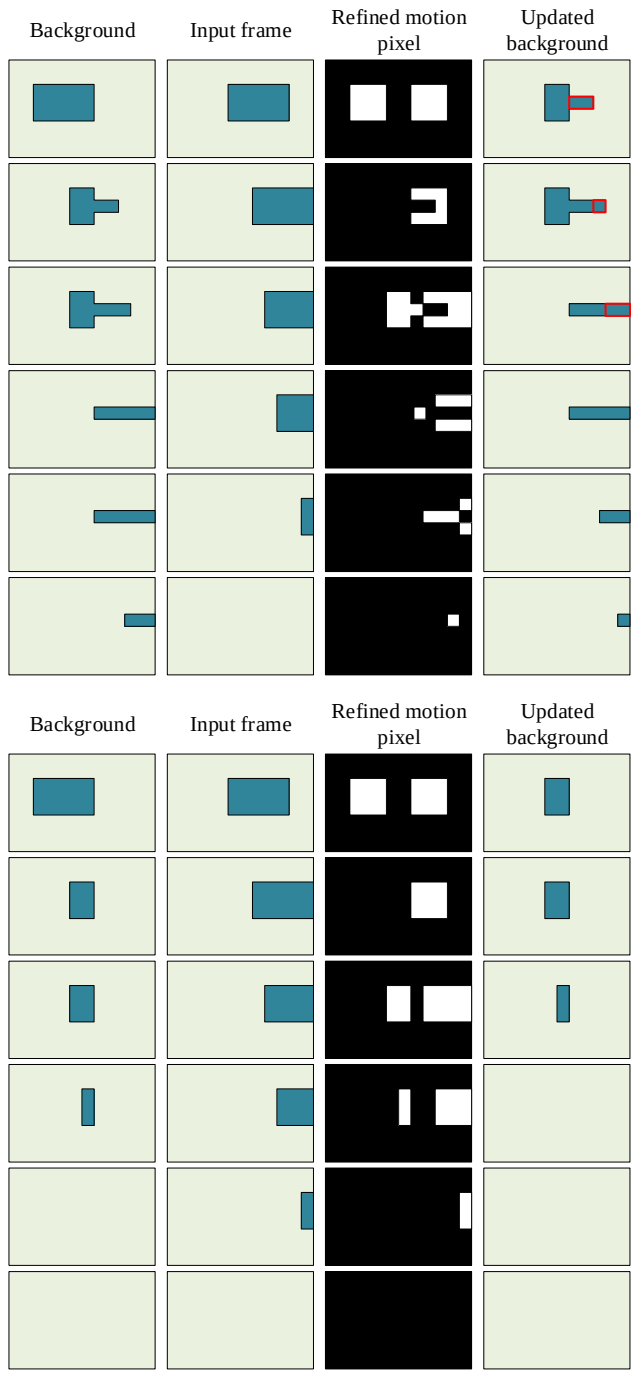


Figure 3.18: Update background image by using the mask (3×3) (top) and (5×5) (bottom) for the case of multipixel shifting motion of objects. Some failed updating pixels when using (3×3) are highlighted.

tal evaluations hereafter investigate the effect of the window size on the accuracy and processing time of foreground detection. After acquiring the updated background image, NIC algorithm enters the new background image to the foreground extraction stage and continuously processes with $(i+1)^{th}$ frame.

3.4 Foreground Detection

In this stage, a binary foreground mask is simply detected based on a subtraction operator with an adaptive threshold. In particular, the difference image between the current input frame F_i and the updated background image B_i is calculated by re-using (3.23):

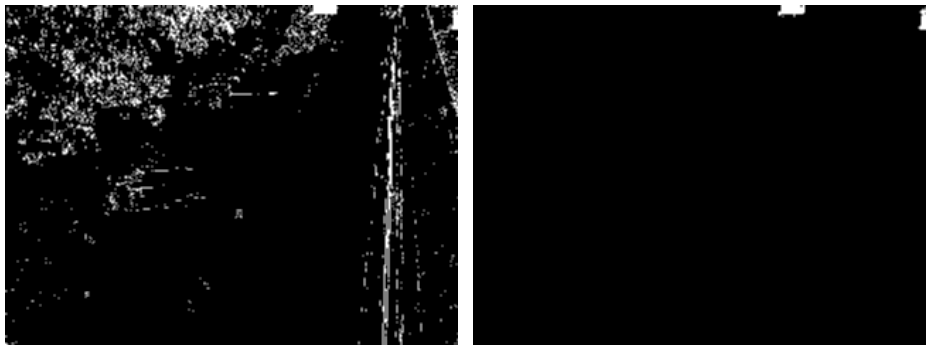
$$Fg_i(x,y) = |F_i(x,y) - B_i(x,y)| \quad ;\forall i \geq 2 \tag{3.32}$$

The difference image FG_i is then segmented to a binary foreground mask by an optimum threshold, identified by the Otsu method [81]. In general, Otsu method is formulated to perform clustering-based image thresholding for segmentation when two classes (foreground pixels and background pixels) are assumed to be sufficiently distinguishable. The optimal threshold aims to effectively separate an image into a background area, denoted as G_0, and a foreground area, denoted as G_1, with a minimum error of pixel detection through minimizing the intra-class variance of the background and foreground class. The threshold, which is exhaustively sought, minimizes the weighted sum of the variance of the two classes:

$$\sigma_\omega^2(g) = \omega_{G_0}(g)\,\sigma_{G_0}^2(g) + \omega_{G_1}(g)\,\sigma_{G_1}^2(g) \tag{3.33}$$

where $\omega_{G_0}(g)$ and $\omega_{G_0}(g)$ are the class probabilities for the intensity g.

$$\begin{aligned} \omega_{G_0}(g) &= \sum_{i=0}^{g-1} p(i) \\ \omega_{G_1}(g) &= \sum_{i=g}^{\max(i)} p(i) \end{aligned} \tag{3.34}$$



(a) (b)

Figure 3.19: An illustration of foreground segmentation using 3.38: (a) τ_{Otsu}, (b) $\max(\tau_{Otsu}, \tau)$.

with $p(i)$ as the probability density function of pixel at the intensity i. The two corresponding pixel classes, $\sigma^2_{G_0}$ and $\sigma^2_{G_1}$ are the individual class variances calculated as follows:

$$\begin{aligned} \sigma^2_{G_0}(g) &= \sum_{i=0}^{g-1} \left((i - \mu_{G_0}(g))^2 \tfrac{p(i)}{\omega_{G_0}(g)} \right) \\ \sigma^2_{G_1}(g) &= \sum_{i=g}^{\max(i)} \left((i - \mu_{G_1}(g))^2 \tfrac{p(i)}{\omega_{G_1}(g)} \right) \end{aligned} \tag{3.35}$$

where the means of corresponding classes are defined as:

$$\begin{aligned} \mu_{G_0}(g) &= \sum_{i=0}^{g-1} \tfrac{i \times p(i)}{\omega_{G_0}(g)} \\ \mu_{G_1}(g) &= \sum_{i=g}^{\max(i)} \tfrac{i \times p(i)}{\omega_{G_1}(g)} \end{aligned} \tag{3.36}$$

The threshold is defined as:

$$\tau_{Otsu} = \arg\min_{g} \left(\sigma^2_{\omega}(g) \right) \tag{3.37}$$

The thresholding process is done with τ_{opt} :

$$Fg^*_i(x,y) = \begin{cases} 1 & ; \forall Fg_i(x,y) \geq \max(\tau_{Otsu}, \tau) \\ 0 & ; otherwise \end{cases} \tag{3.38}$$

(a) (b) (c)

Figure 3.20: An illustration of post-processing: (a) extracted foreground by Otsu thresholding, (b) foreground after morphological closing operator, and (c) final foreground after filling holes on (b).

In (3.38), we reuse the constant threshold τ in 3.24 for final segmentation. In the outdoor environment, artifacts usually appear in a scene either besides moving objects or alone. The artifacts can be background movements such as water wave, leaf motion, lightning reflection, and etc. It should be noted that NIC algorithm did not update the intensity for these artifacts by eliminating them out of the set of refined motion pixels, they are, therefore, potentially detected by Otsu threshold in the case of non-object frames. To handle this issue, we set up a ground threshold which equals to the constant threshold τ to be sure that the foreground overpass artifact motion in non-object frames as illustrated in Figure 3.19.

The primitive foreground sometimes consists of disconnected edges because of drastic luminance changes in dynamic scenes. During the post-processing, some morphological operations [82] are applied to fuse narrow breaks, eliminate holes, and fill gaps of contour. Concretely, we first perform morphological closing and then fills holes as an example illustrated in Figure 3.20.

Chapter 4

Experimental Results and Discussion

In this chapter, we evaluate the performance of NIC algorithm for moving object detection in terms of accuracy and complexity. It should be noted that in the field of foreground detection, the quality of background image estimation algorithm is fundamentally evaluated by the subsequent foreground detection results. The reasons are: (i) the ground truth for background estimation or modeling is hard to be defined in the real world, i.e. a pure static scene does not exist, and (ii) most background models are highly abstract and are treated as an intermediate step for detection.

4.1 Experiment Setup

As a default parameter set up, the constant threshold $\tau = 20$ and the mask of size 7×7 are fixed. The values of these parameters are chosen based on parameter sensitivity investigation to achieve a balance between detection accuracy and computational complexity. Four experiments performed for evaluation of NIC algorithm are described as follows:

- First experiment: evaluates the performance of moving object detection in the terms of accuracy and processing speed. In details, we at first investigate the influence of NIC parameters on moving object detection efficiency for the best score selection. Secondly, the qualitative and quantitative results of the proposed algorithm with the best core parameters are reported. The processing speed is also evaluated in this experiment on various video resolutions.

- Second experiment: compares the moving object detection performance between the our proposed algorithm with other existing methods.

4.1.1 Dataset

In order to benchmark foreground detection performance, we utilized ChangeDetection 2014, an available dataset, which provides realistic and camera-captured videos representing a wide range of detection challenges. ChangeDetection covers indoor and outdoor visual data recorded for surveillance and smart environment scenarios. In particular, totally 10 videos representing five common background challenges such as baseline, dynamic background, camera jitter, intermittent object motion, and challenging weather, are selected for evaluating NIC algorithm.

Baseline: This group represents an intermixture of four mild challenges including dynamic background, camera jitter, intermittent object motion, and shadows. In general, most of the videos in this group are fairly easy, but not negligible, therefore they are mainly treated as the reference for the benchmark.

- *Highway*: presents subtle background motion and isolated shadows with objects moving in and moving out of the scene sequentially.
- *Office*: captures an object moving in, stop for a while, and moving out of the scene in the indoor environment.

Dynamic Background: This category contains outdoor scenes with strong and incessant background motion.

- *Canoe*: represents boat events on the shimmering water with successive background illuminance change due to sunlight reflection.
- *Overpass*: is characterized by strong and incessant tree shaken caused by the wind.

Camera Jitter: This category contains indoor and outdoor video recorded by vibrating cameras with different jitter magnitude.

- *Badminton*: presents multi-object movement in an unstable capturing condition.
- *Traffic*: records traffic activities in the outdoor environment with strong vibration and object shadows.

Table 4.1: The information of number of frame and resolution of all testing videos

Video	Number of frame	Resolution
highway	1700	320×240
office	2050	360×240
canoe	1189	320×240
overpass	3000	320×240
badminton	1150	720×480
traffic	1570	320×240
sofa	2750	320×240
parking	2500	320×240
skating	3900	540×360
blizzard	7000	720×480

Intermittent Object Motion: This group covers several sequences where "ghosting" artifacts in detection motion are illustrated,

- *Parking*: contains suddenly moving objects in a scene, e.g. a parked vehicle driving away.
- *Sofa*: describes an event of abandoned objects, i.e. objects move in, stop for short and long-term durations, and move out of a scene.

Challenging Weather: This category includes outdoor videos recorded in challenging winter weather conditions.

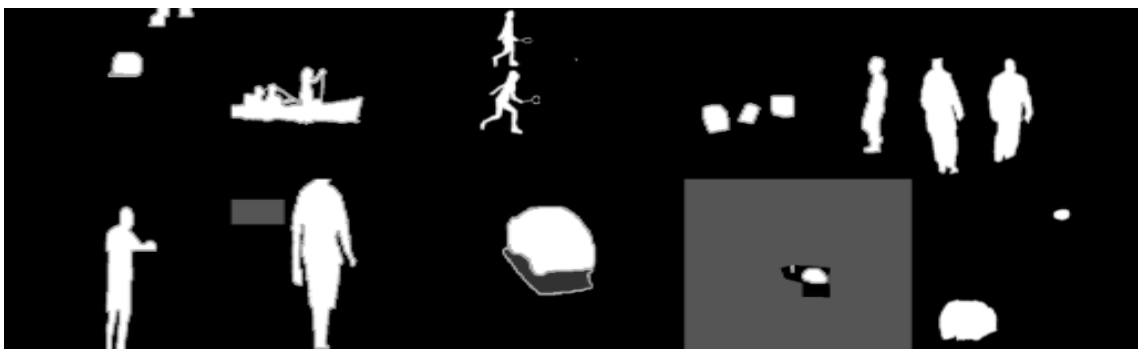
- *Blizzard*: presents outdoor traffic activity where snow storm and fog densely appear.
- *Skating*: describes skating activity with thick snow on the ground and heavy snow falling.

The information about number of frame and image resolution of all testing videos are listed in Table 4.1. Some input samples of videos are shown in Figure 4.1(a). It should be noted that the accurate ground truths as shown in Figure 4.1(b) are also provided for all frame of all videos for testing and evaluation.

4.1.2 Evaluation Metric

All experiments are performed in Matlab environment using a laptop which is equipped by a 2.67 GHz Intel Core i7 CPU and 16GB RAM. For evaluating the foreground detection performance, totally seven quantitative metrics are calculated based on comparing the achieved results with

(a)



(b)

Figure 4.1: Samples of testing videos used for evaluation: (a) first row: highway, canoe, badminton, sofa, and skating; second row: office, overpass, traffic, parking, and blizzard. (b) the corresponding ground truth images.

ground truths. They include Recall (Re) or True Positive Rate, Specificity (Sp) or True Negative Rate, False Positive Rate (FPR), False Negative Rate (FNR), Percentage of Wrong Classification (PWC), Precision (Pre) or Positive Predictive Value, and F-measure or F1 score. Among all quantitative metrics, FPR, FNR, and PWC are smaller as better while remaining metrics as Re, Sp, Pre, F1 are higher as better. To benchmark the computational complexity of NIC algorithm, the average processing speed measured by frame per second (FPS) metric is recommended for utilization.

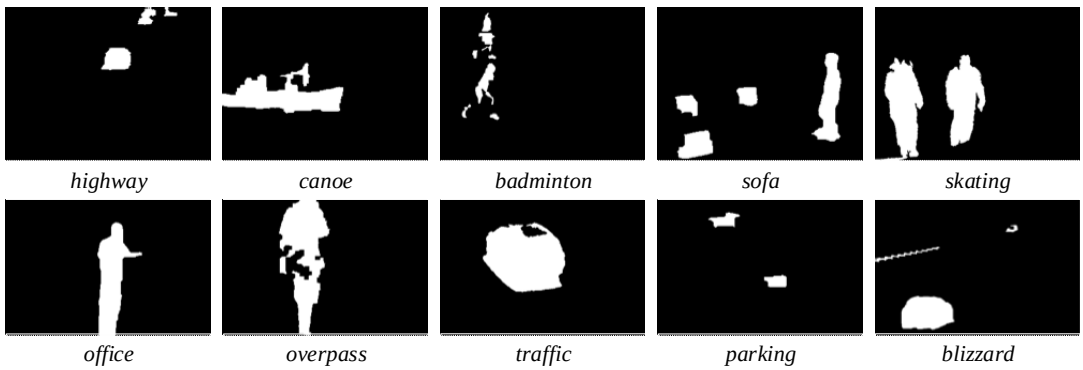


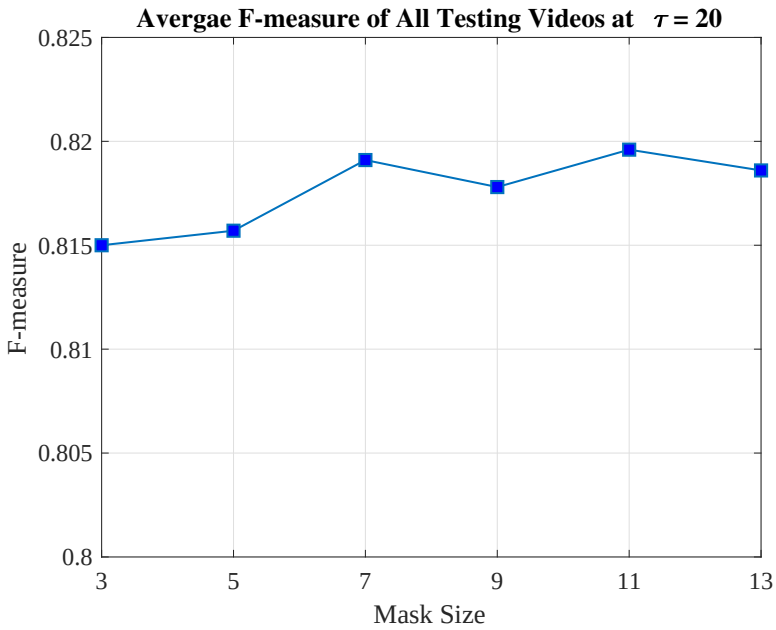
Figure 4.2: Some examples of foreground result detected by NIC algorithm.

4.2 First Experiment: Moving Object Detection

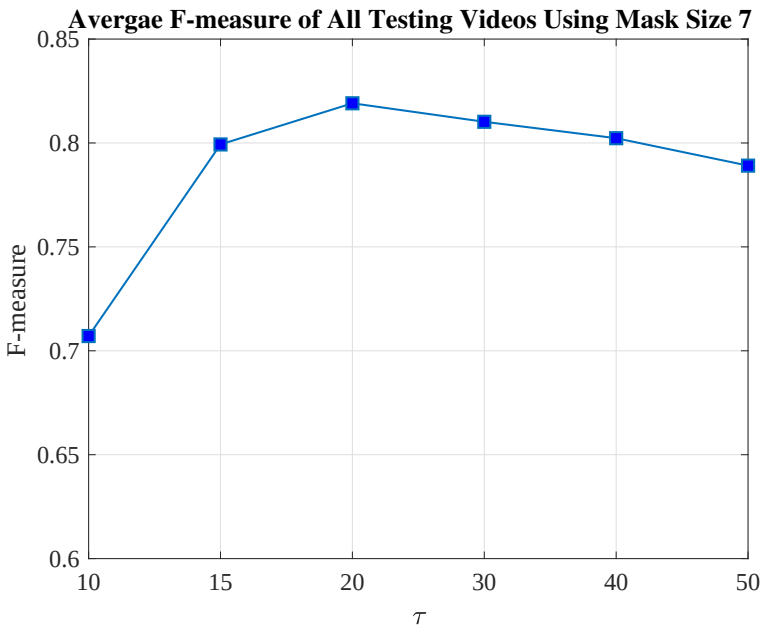
4.2.1 Detection Accuracy

At first, the visual foreground detection results of all testing videos which are given in Figure 4.2 obtained under the default parameter configuration, e.g. $\tau = 20$ and mask size 7×7. It should be noted that, we randomly select frames in corresponding videos for illustration, for instance, frame $\#0638$ of *highway*, $\#0995$ of *office*, $\#0915$ of *canoe*, $\#2395$ of *overpass*, $\#0866$ of *badminton*, $\#1208$ of *traffic*, $\#1906$ of *sofa*, $\#1465$ of *parking*, $\#1870$ of *skating*, and $\#1750$ of *blizzard*. Compared with ground truth, the proposed method mostly succeeds in detecting foreground. The worse results can be found in *badminton*, *sofa*, and *parking*. In *badminton*, the objects are seriously fragmented by strong vibration of a camera. Due to an operating characteristic of NIC that updates non-moving objects, which staying in a scene for a long time, to the background class, the proposed method is unable to detect non-moving objects in the background image. With *sofa* sample, a box is perceived as the background class by NIC algorithm. When an input frame without a box in the scene is provided, the box is even detected as foreground based on the difference between the input frame and the background image. NIC algorithm is able to fix this problem by consecutively updating background image based on the input frames, although the process can take time to absolutely discard abandoned objects.

We investigate the influence of NIC parameters including the constant threshold τ and the mask size to the overall accuracy of foreground detection. The purpose of this experiment is to

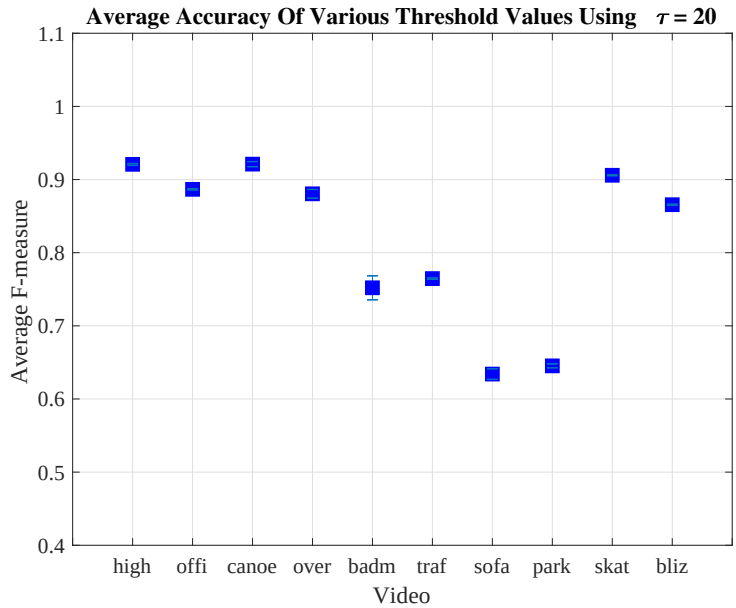


(a)

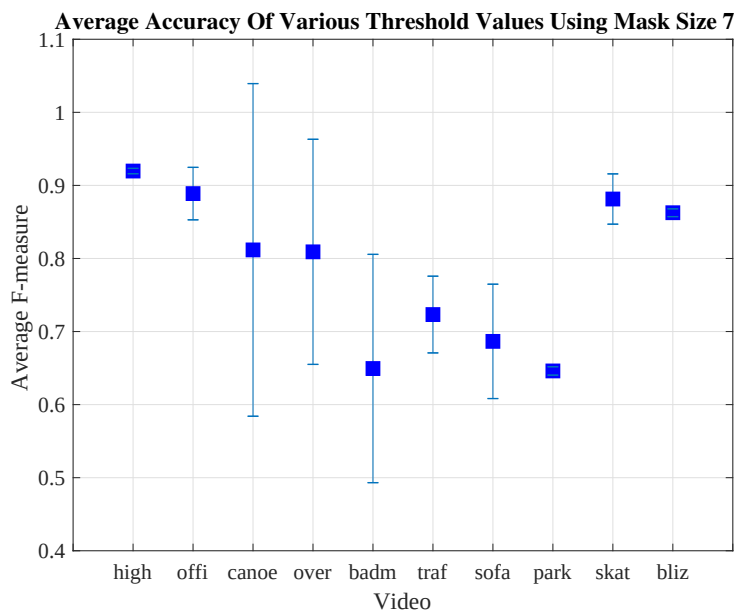


(b)

Figure 4.3: Average F-measure of foreground detection under various parameter configuration: (a) $\tau = 20$ and mask size $\{3, 5, 7, 9, 11, 13\}$, (b) $\tau = \{10, 15, 20, 30, 40, 50\}$ and mask size 7.



(a)



(b)

Figure 4.4: Average F-measure with standard deviation of foreground detection on each particular video sample to evaluate (a) the influence of mask size with $\tau = 20$, (b) the influence of τ with mask size 7.

Table 4.2: The foreground detection accuracy under various mask size values

Mask Size	Re	Sp	FPR	FNR	PWC	Pre	F1	Re	Sp	FPR	FNR	PWC	Pre	F1
	highway							**office**						
3	0.909	0.996	0.004	0.091	0.949	0.929	0.919	0.886	0.992	0.009	0.114	1.575	0.886	0.886
5	0.910	0.996	0.004	0.090	0.944	0.929	0.920	0.886	0.992	0.009	0.114	1.583	0.885	0.885
7	0.914	0.996	0.004	0.086	0.922	0.929	0.922	0.889	0.991	0.009	0.111	1.564	0.885	0.887
9	0.914	0.996	0.004	0.086	0.924	0.929	0.921	0.889	0.991	0.009	0.111	1.560	0.885	0.887
11	0.914	0.996	0.004	0.086	0.921	0.929	0.922	0.892	0.991	0.009	0.109	1.560	0.884	0.888
13	0.914	0.996	0.004	0.086	0.921	0.929	0.922	0.890	0.991	0.009	0.110	1.569	0.884	0.887
	canoe							**overpass**						
3	0.918	0.997	0.003	0.082	0.592	0.915	0.917	0.799	1.000	0.000	0.201	0.305	0.968	0.875
5	0.916	0.997	0.003	0.084	0.572	0.922	0.919	0.800	1.000	0.000	0.200	0.303	0.969	0.876
7	0.916	0.998	0.002	0.084	0.514	0.937	0.927	0.809	1.000	0.000	0.191	0.294	0.966	0.881
9	0.916	0.997	0.003	0.084	0.540	0.931	0.923	0.801	1.000	0.000	0.199	0.303	0.968	0.876
11	0.920	0.997	0.003	0.080	0.563	0.921	0.921	0.827	1.000	0.000	0.174	0.274	0.964	0.890
13	0.920	0.997	0.003	0.080	0.561	0.922	0.921	0.817	1.000	0.000	0.183	0.285	0.965	0.885
	badminton							**traffic**						
3	0.801	0.986	0.014	0.200	2.041	0.669	0.729	0.749	0.986	0.014	0.251	2.877	0.780	0.764
5	0.799	0.987	0.013	0.201	1.981	0.679	0.735	0.751	0.986	0.014	0.249	2.866	0.780	0.765
7	0.778	0.991	0.009	0.222	1.641	0.752	0.765	0.749	0.986	0.014	0.251	2.877	0.780	0.764
9	0.799	0.989	0.011	0.201	1.786	0.714	0.754	0.751	0.986	0.014	0.249	2.865	0.780	0.766
11	0.795	0.990	0.010	0.205	1.651	0.742	0.768	0.748	0.986	0.014	0.252	2.883	0.780	0.764
13	0.797	0.990	0.011	0.203	1.709	0.729	0.762	0.750	0.986	0.014	0.250	2.873	0.780	0.765
	sofa							**parking**						
3	0.642	0.984	0.016	0.358	3.107	0.645	0.643	0.775	0.947	0.053	0.225	6.642	0.550	0.644
5	0.642	0.984	0.016	0.358	3.137	0.641	0.641	0.777	0.947	0.053	0.223	6.633	0.550	0.644
7	0.654	0.981	0.019	0.346	3.342	0.609	0.631	0.771	0.948	0.052	0.229	6.600	0.553	0.644
9	0.653	0.982	0.018	0.347	3.264	0.620	0.636	0.774	0.947	0.053	0.226	6.646	0.550	0.643
11	0.653	0.980	0.020	0.347	3.401	0.602	0.627	0.783	0.947	0.053	0.217	6.550	0.554	0.649
13	0.652	0.980	0.020	0.348	3.394	0.603	0.627	0.780	0.948	0.053	0.220	6.543	0.555	0.648
	skating							**blizzard**						
3	0.916	0.995	0.006	0.085	0.942	0.897	0.906	0.802	0.999	0.001	0.198	0.288	0.942	0.867
5	0.916	0.995	0.006	0.085	0.940	0.897	0.906	0.803	0.999	0.001	0.197	0.290	0.940	0.866
7	0.919	0.994	0.006	0.081	0.932	0.896	0.907	0.802	0.999	0.001	0.198	0.291	0.939	0.865
9	0.916	0.994	0.006	0.084	0.946	0.896	0.905	0.803	0.999	0.001	0.197	0.290	0.939	0.866
11	0.916	0.994	0.006	0.084	0.951	0.895	0.905	0.802	0.999	0.001	0.198	0.292	0.938	0.865
13	0.916	0.995	0.006	0.084	0.944	0.896	0.906	0.802	0.999	0.001	0.198	0.292	0.939	0.865

evaluate the robustness of NIC algorithm under various parameter configurations and to report the best average score for the default setting so far. The investigation is done for all testing video samples and the results are reported using seven quantitative metrics.

According to $\tau = 20$, Table 4.2 report the detection accuracy results of all videos in details under various mask sizes. The results of F-measure metric are summarized in Table 4.4 and the average results are plotted in Figure 4.3(a). For most of the video samples, the results vary insignificantly by the mask size parameter. It can be observed that the computed standard deviation of F-measure corresponding to each particular mask size is really small as Figure 4.4(a). For a

Table 4.3: The foreground detection accuracy under various values of the constant threshold τ

τ	Re	Sp	FPR	FNR	PWC	Pre	F1	Re	Sp	FPR	FNR	PWC	Pre	F1
	highway							**office**						
10	0.896	0.996	0.004	0.104	1.020	0.930	0.912	0.807	0.990	0.010	0.193	2.270	0.856	0.831
15	0.910	0.996	0.004	0.090	0.943	0.929	0.920	0.858	0.991	0.009	0.142	1.843	0.873	0.865
20	0.914	0.996	0.004	0.086	0.922	0.929	0.922	0.889	0.991	0.009	0.111	1.564	0.885	0.887
30	0.915	0.996	0.004	0.085	0.920	0.929	0.922	0.919	0.992	0.008	0.081	1.295	0.896	0.907
40	0.915	0.996	0.004	0.085	0.924	0.928	0.921	0.932	0.992	0.008	0.068	1.196	0.899	0.915
50	0.915	0.996	0.004	0.085	0.927	0.928	0.921	0.953	0.992	0.008	0.047	1.030	0.903	0.927
	canoe							**overpass**						
10	0.214	1.000	0.000	0.786	2.830	0.941	0.349	0.339	1.000	0.000	0.661	0.898	0.975	0.503
15	0.793	0.999	0.001	0.207	0.876	0.952	0.865	0.680	1.000	0.000	0.320	0.454	0.974	0.801
20	0.916	0.998	0.002	0.084	0.514	0.937	0.927	0.809	1.000	0.000	0.191	0.294	0.966	0.881
30	0.921	0.997	0.003	0.079	0.587	0.914	0.917	0.830	1.000	0.000	0.170	0.276	0.958	0.889
40	0.923	0.996	0.004	0.077	0.643	0.898	0.910	0.838	0.999	0.001	0.162	0.273	0.953	0.892
50	0.924	0.995	0.005	0.076	0.714	0.881	0.902	0.841	0.999	0.001	0.159	0.281	0.943	0.889
	badminton							**traffic**						
10	0.725	0.996	0.004	0.275	1.287	0.878	0.794	0.552	0.989	0.011	0.448	3.791	0.774	0.644
15	0.747	0.994	0.006	0.253	1.440	0.817	0.781	0.647	0.988	0.012	0.353	3.319	0.782	0.708
20	0.778	0.991	0.009	0.222	1.641	0.752	0.765	0.749	0.986	0.014	0.251	2.877	0.780	0.764
30	0.799	0.973	0.027	0.201	3.345	0.508	0.621	0.815	0.984	0.016	0.185	2.682	0.768	0.791
40	0.810	0.952	0.048	0.190	5.294	0.374	0.512	0.832	0.972	0.028	0.168	3.679	0.663	0.738
50	0.814	0.928	0.072	0.186	7.575	0.287	0.424	0.831	0.963	0.037	0.169	4.561	0.596	0.694
	sofa							**parking**						
10	0.639	0.984	0.016	0.361	3.139	0.641	0.640	0.746	0.956	0.044	0.254	6.029	0.587	0.657
15	0.640	0.982	0.018	0.360	3.256	0.624	0.632	0.766	0.950	0.050	0.234	6.417	0.562	0.649
20	0.654	0.981	0.019	0.346	3.342	0.609	0.631	0.771	0.948	0.052	0.229	6.600	0.553	0.644
30	0.678	0.981	0.019	0.322	3.204	0.622	0.649	0.777	0.946	0.054	0.223	6.685	0.548	0.643
40	0.832	0.983	0.017	0.168	2.382	0.688	0.753	0.777	0.946	0.054	0.223	6.687	0.548	0.642
50	0.860	0.989	0.011	0.140	1.707	0.774	0.815	0.777	0.946	0.054	0.223	6.688	0.548	0.642
	skating							**blizzard**						
10	0.801	0.998	0.002	0.199	1.184	0.952	0.870	0.803	1.000	0.000	0.197	0.278	0.951	0.871
15	0.914	0.995	0.005	0.086	0.926	0.900	0.907	0.802	0.999	0.001	0.198	0.307	0.925	0.859
20	0.919	0.994	0.006	0.081	0.932	0.896	0.907	0.802	0.999	0.001	0.198	0.291	0.939	0.865
30	0.916	0.994	0.006	0.084	0.958	0.893	0.904	0.801	0.999	0.001	0.199	0.309	0.924	0.858
40	0.918	0.991	0.009	0.082	1.229	0.846	0.881	0.801	0.999	0.001	0.199	0.309	0.924	0.858
50	0.919	0.983	0.017	0.081	2.016	0.738	0.819	0.801	0.999	0.001	0.199	0.309	0.923	0.858

particular video sample, a most efficient mask size is determined. For instance, 13×13 is the best choice for *highway*, however, its accuracy margin is very small, approximately 10^{-4} of F-measure, if compared with 9×9, the runner-up candidate. With high-speed motions in *highway*, camera vibration in *badminton*, and heavily incessant motion background in *canoe*, *overpass*, *blizzard*, a larger mask can become a satisfactory choice. On the other hand, small and medium sizes of the mask are more suitable for slow moving objects. It should be reminded that a larger size is probably to increase the computational complexity of standard deviation calculation.

By fixing the mask 7×7 as a balancing choice between accuracy and complexity, we validate

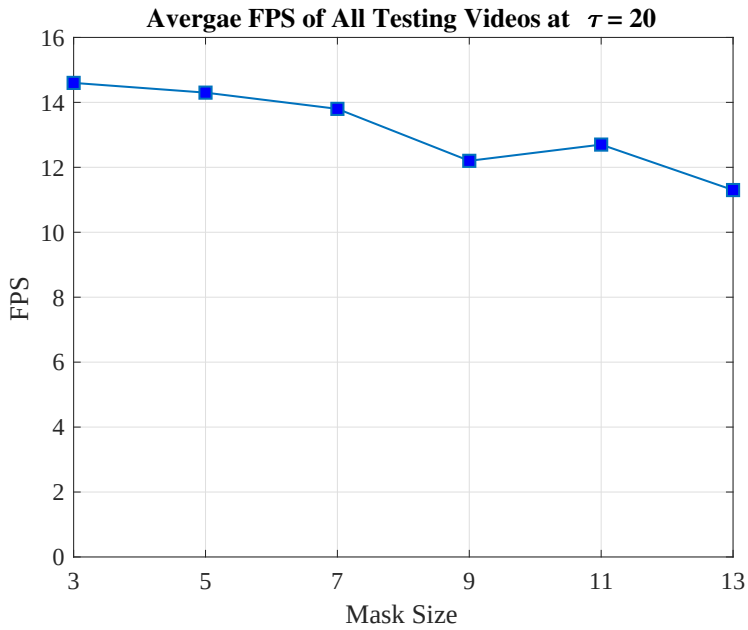
Table 4.4: F-measure results of testing videos under various mask sizes

Video	**Mask Size**						**Threshold τ**					
	3	5	7	9	11	13	10	15	20	30	40	50
highway	0.919	0.920	0.922	0.921	0.922	0.922	0.912	0.920	0.922	0.922	0.921	0.921
office	0.886	0.885	0.887	0.887	0.888	0.887	0.831	0.865	0.887	0.907	0.915	0.927
canoe	0.917	0.919	0.927	0.923	0.921	0.921	0.349	0.865	0.927	0.917	0.910	0.902
overpass	0.875	0.876	0.881	0.876	0.890	0.885	0.503	0.801	0.881	0.889	0.892	0.889
badminton	0.729	0.735	0.765	0.754	0.768	0.762	0.794	0.781	0.765	0.621	0.512	0.424
traffic	0.764	0.765	0.764	0.766	0.764	0.765	0.644	0.708	0.764	0.791	0.738	0.694
sofa	0.643	0.641	0.631	0.636	0.627	0.627	0.640	0.632	0.631	0.649	0.753	0.815
parking	0.644	0.644	0.644	0.643	0.649	0.648	0.657	0.649	0.644	0.643	0.642	0.642
skating	0.906	0.906	0.907	0.905	0.905	0.906	0.870	0.907	0.907	0.904	0.881	0.819
blizzard	0.867	0.866	0.865	0.866	0.865	0.865	0.871	0.865	0.865	0.858	0.858	0.858

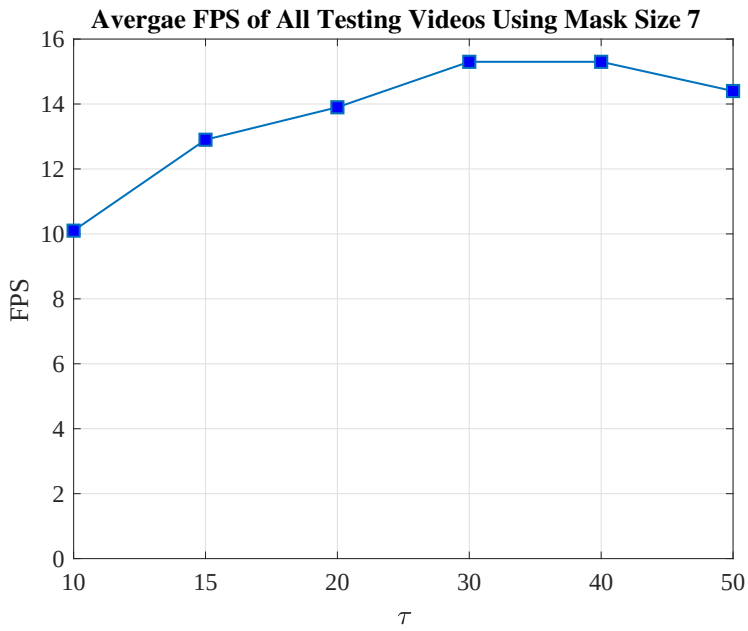
the impact of the constant threshold τ on the overall accuracy of foreground detection by sweeping the value from 10 to 50. The detail results are reported in Table 4.3 for all metrics. The F-measure outcomes are summarized in Table 4.4 and outlined in average in Figure 4.3(b). Compared with the mask size, the constant threshold has a more sensitive effect on detection accuracy. This can be realized by estimating the standard deviation of F-measure plotted in Figure 4.4(b). With two samples in category of dynamic background, the variation of results is significant, e.g. the F-measure increases from 0.3491 to 0.8652 in *canoe* and from 0.5029 to 0.8007 in *overpass* when increasing τ from 10 to 15. Besides that, some samples in the category of camera jitter are also drastically influenced by this parameter. With a small τ, there are so many pixels including moving objects and noise which are requested to correct the intensity by NIC algorithm. It can be seen that updating noise pixels causes the error of updating pixel intensity of surrounding motion pixels that leads the failure in updating of the background image, especially with videos containing much more background motions as *canoe* and *overpass*. In the case of $\tau \geq 40$, the overall accuracy is tendentiously decreased because some important motion pixels are ignored. As a recommendation, τ should be chosen in the range $[15, 30]$ to eliminate noise pixels while still keeping enough motion pixels for updating.

4.2.2 Processing Speed

Secondly, the computational complexity of NIC algorithm is evaluated and analyzed on the processing time, calculated by frame per second metric. In details, a profiling tool of Matlab is utilized to measure the time required for the whole method of background subtraction. The algorithm com-



(a)



(b)

Figure 4.5: Average FPS achieved under various parameter configuration: (a) $\tau = 20$ and mask size $\{3, 5, 7, 9, 11, 13\}$, (b) $\tau = \{10, 15, 20, 30, 40, 50\}$ and mask size 7.

Table 4.5: Average processing speed (FPS) under various mask sizes

Video	**Mask Size**						**Threshold τ**					
	3	5	7	9	11	13	10	15	20	30	40	50
highway	18.3	19.1	15.9	15.5	15.5	16.3	14.7	17.4	15.9	17.5	16.9	15.2
office	17.2	15.5	17.5	15.1	14.9	13.7	15.1	17.0	17.5	23.2	18.8	14.2
canoe	06.7	06.2	06.5	06.7	06.5	06.3	02.9	05.5	06.5	07.0	07.1	07.2
overpass	14.1	12.7	10.6	09.0	09.3	08.9	03.4	07.1	10.6	15.0	15.5	15.2
badminton	04.0	04.1	04.0	04.2	04.0	03.9	01.4	02.0	04.0	02.8	03.0	02.8
traffic	03.2	03.0	02.9	02.8	02.9	02.7	01.9	03.9	02.9	07.9	11.1	10.2
sofa	21.3	22.5	21.7	21.2	21.8	20.1	22.0	23.3	21.7	25.2	25.3	26.3
parking	43.1	41.7	41.0	29.1	33.8	24.5	23.5	35.1	41.0	36.5	37.1	34.6
skating	03.1	03.1	03.3	03.2	03.2	02.9	01.5	04.2	03.7	03.3	03.2	03.1
blizzard	14.5	15.2	14.9	14.9	14.8	13.7	14.7	13.9	14.9	14.9	15.1	14.9

plexity is examined under various mask sizes and threshold values τ with the FPS results reported in Table 4.5. Generally, it is seen that larger frames (or higher resolution) need more time for processing. Besides that, the processing speed of NIC algorithm also depends on the density of moving objects in a scene. With the same resolution of 320×240, *traffic* takes more time for intensity updating than *highway* due to (i) the greater density of moving objects in a scene and (ii) the motion pixels from camera vibration. As already discussed in chapter 3, a smaller mask size requires less time for calculating standard deviation metric in the updating process, that means, the method detects foreground faster with a smaller mask. Based on the results shown in Table 4.5, the FPS goes down along the increment of mask size for all testing videos. This thing is easily observed in Figure 4.5(a). In the contrast, the processing speed is significantly improved if increase the value of threshold τ. It is seen that τ disturbs extremely to the algorithm performance. For instance, *skating* achieves 1.5 FPS at $\tau = 10$ and 4.2 FPS at $\tau = 15$. The FPS of *canoe* is raised from 2.9 at $\tau = 10$ to 6.5 at $\tau = 20$ and be stable around $\sim$7.0 FPS at $\tau \geq 30$. Similarly, a noticeable improvement also occurs in *overpass*. It can be realized that those samples have much more background motions than others. If τ is set at a small value, noise pixels representing background motions are identified and refined as motion pixels for intensity updating process. Thus, as a result, NIC algorithm takes more time for updating the background image undesirably. A small threshold cannot guarantee a high performance in the terms of detection accuracy and processing speed. Based on the average result shown in Figure 4.5(b), the FPS enhancement is so remarkable in the range $\tau = [20, 30]$. In summary, based on the thoughtful analysis, we recommend the parameter configuration for NIC algorithm as follows the mask 7×7 and the constant threshold $\tau = [20, 40]$.

Table 4.6: Parameter configuration of comparing methods.

Method	Parameter Configuration
Average Filtering	$N = 25$, $\upsilon = 1$ FPS
Median Filtering	$N = 25$, $\upsilon = 1$ FPS
Running Average	$\alpha = 0.05$ (selective mode)
Approximated Median	None
Running Gaussian Average	$\alpha = 0.05$ (selective mode)
Histogram Over Time	$N = 25$, $\upsilon = 1$ FPS (selective mode)
NIC	$\tau = 20$, Mask Size 7×7

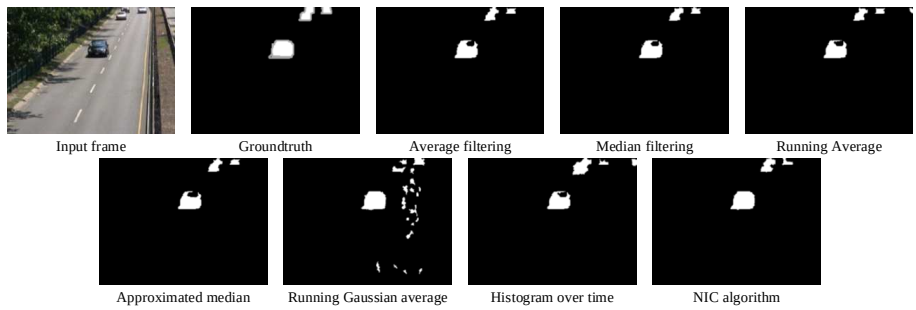
4.3 Second Experiment: Method Performance Comparison

In this section, we compare the moving object detection performance, including the accuracy and processing speed, of the proposed NIC algorithm and other methods which are divided into two classes: background reference image based and background model based approaches.

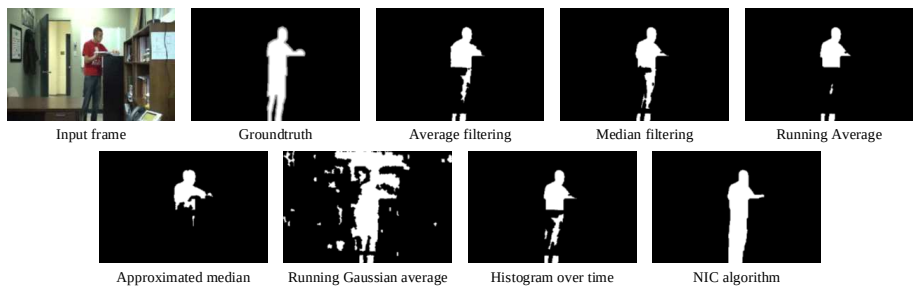
4.3.1 Comparing with Background Image Estimation Methods

We compare NIC algorithm with other background reference image based methods such as average/median filtering, running average [15], approximated median [19], running Gaussian average [29], and histogram over time [23]. These are recognized as the baseline algorithms in the field where the background of a scene is modeled by a reference image. The method parameters are configured as in Table 4.6. For saving the computation cost, the buffer of the filtering approaches has the size $N = 25$ frames and the buffer sampling rate $\upsilon = 1$ FPS. Similarly, the histogram is updated with sampling rate $\upsilon = 1$ FPS to significantly reduce the processing time of the histogram over time method. The advantages of this group are simple implementation and fast processing speed, however, the accuracy is quite poor, especially with some challenging background scenarios such as dynamic background, camera jitter, and intermittent object motion. The visual results of comparison are shown in Figure 4.6, 4.7, and 4.8 while the quantitative results are reported in Table 4.7 and 4.8. It should be noted that we use the same post-processing for all methods with salt&pepper noise removal, closing operation, and hole filling.

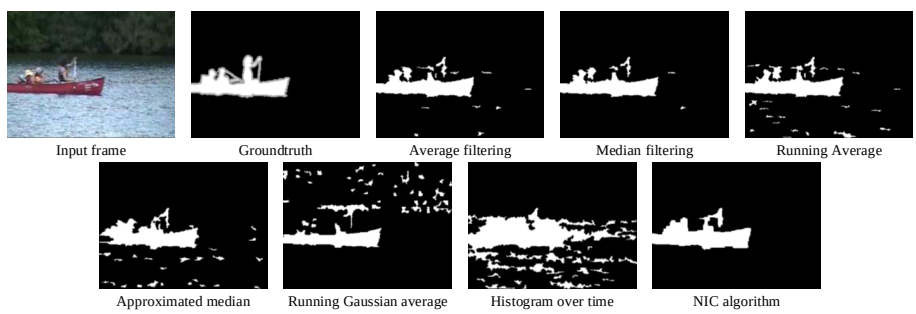
From the numerical results in Table Table 4.7 and 4.8, our proposed algorithm outperformed all of the baseline algorithms. For high challenging samples as *canoe* and *overpass* of dynamic background, *badminton* and *traffic* of camera jitter, *skating* and *blizzard* of bad weather, NIC algo-



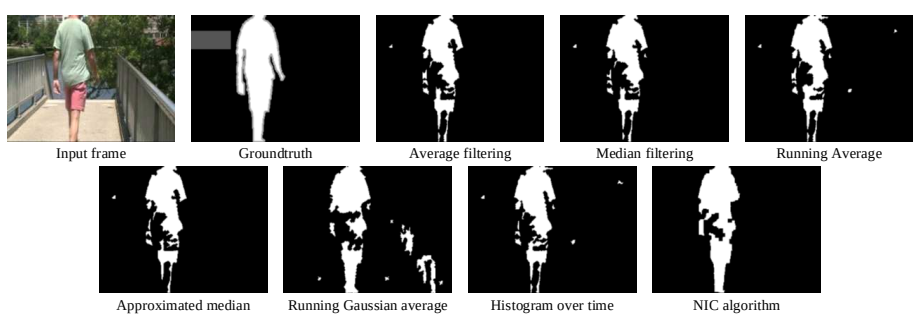
(a)



(b)

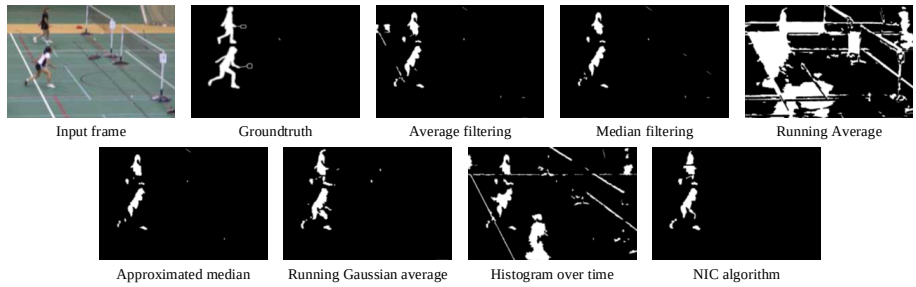


(c)

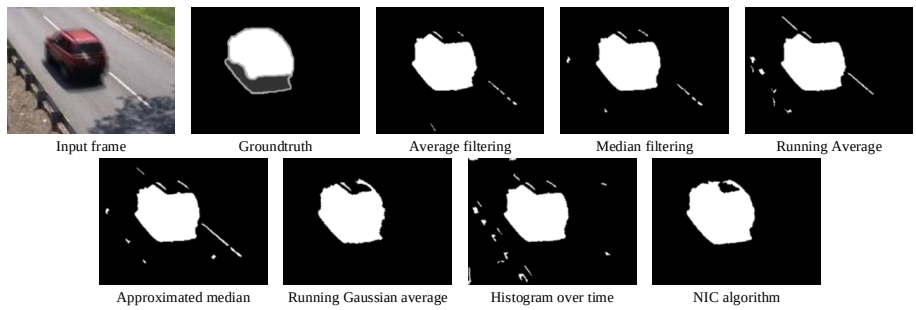


(d)

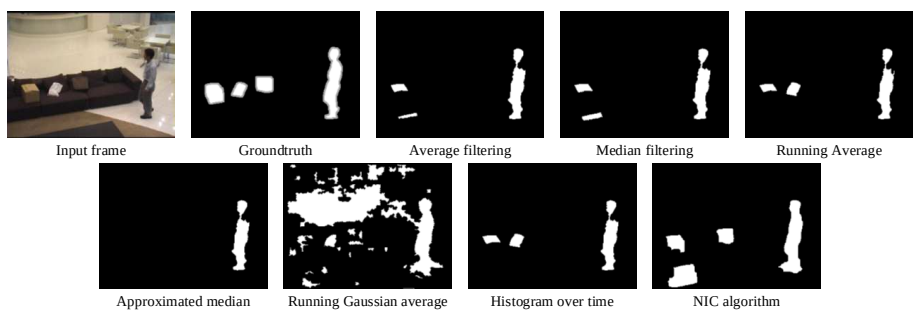
Figure 4.6: Foreground images of NIC and other reference image based background subtraction methods: (a) *highway*, (b) *office*, (c) *canoe*, and (d) *overpass*.



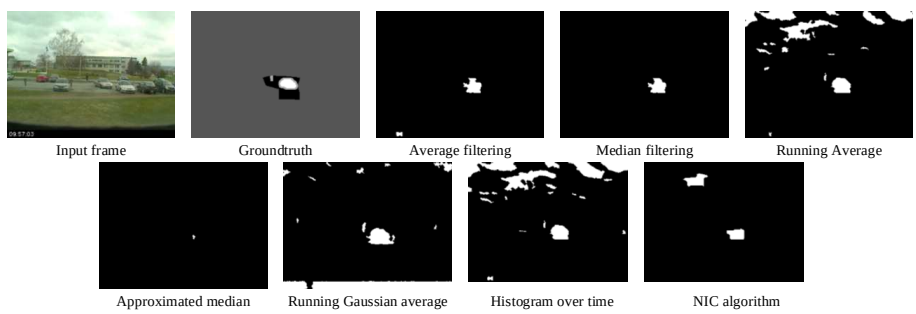
(a)



(b)



(c)



(d)

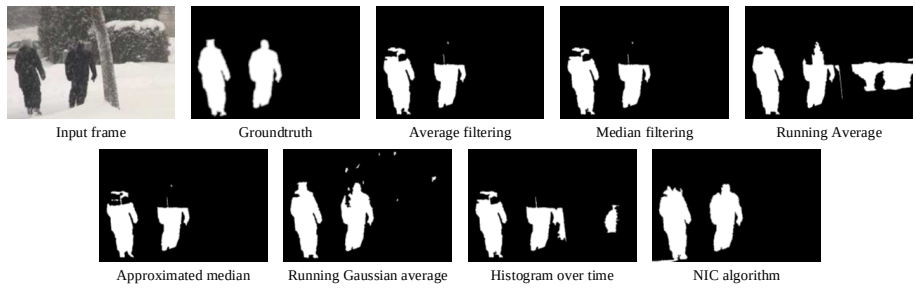
Figure 4.7: Continued: (a) *badminton*, (b) *traffic*, (c) *sofa*, and (d) *parking*.

Table 4.7: Foreground detection accuracy comparison of NIC with several baseline methods of background reference image

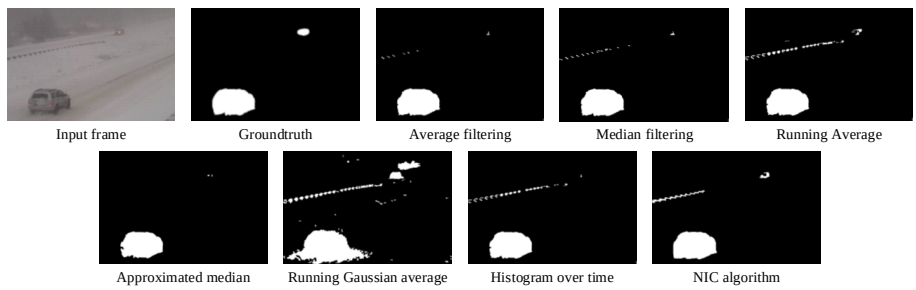
Method	Re	Sp	FPR	FNR	PWC	Pre	F1
highway							
Average Filtering	0.7855	0.9923	0.0077	0.2145	1.9910	0.8660	0.8238
Median Filtering	0.8219	0.9958	0.0042	0.1781	1.4528	0.9246	0.8702
Average Running	0.7939	0.9923	0.0077	0.2061	1.9462	0.8665	0.8286
Approximated Median	0.7790	0.9956	0.0044	0.2210	1.7199	0.9184	0.8429
Running Gassian Average	0.6609	0.9937	0.0063	0.3391	2.5983	0.8692	0.7509
Histogram Over Time	0.7947	0.9908	0.0092	0.2053	2.0863	0.8441	0.8186
NIC Algorithm	0.9140	0.9956	0.0044	0.0860	0.9219	0.9292	0.9216
office							
Average Filtering	0.6050	0.9833	0.0167	0.3950	4.2812	0.7288	0.6612
Median Filtering	0.4539	0.9892	0.0108	0.5461	4.7798	0.7563	0.5673
Average Running	0.5333	0.9945	0.0055	0.4667	3.7380	0.8770	0.6633
Approximated Median	0.3220	0.9894	0.0106	0.6780	5.6641	0.6933	0.4398
Running Gassian Average	0.8024	0.8707	0.1293	0.1976	13.4005	0.3152	0.4526
Histogram Over Time	0.7801	0.9998	0.0002	0.2199	1.5383	0.9962	0.8750
NIC Algorithm	0.8894	0.9914	0.0086	0.1106	1.5639	0.8847	0.8870
canoe							
Average Filtering	0.9179	0.7029	0.2971	0.0821	28.9524	0.1019	0.1834
Median Filtering	0.8882	0.7598	0.2402	0.1118	23.5645	0.1195	0.2107
Average Running	0.8523	0.8602	0.1398	0.1477	14.0053	0.1829	0.3012
Approximated Median	0.7700	0.8037	0.1963	0.2300	19.7468	0.1259	0.2164
Running Gassian Average	0.3754	0.9569	0.0431	0.6246	6.3671	0.2424	0.2946
Histogram Over Time	0.9442	0.6890	0.3110	0.0558	30.1976	0.1003	0.1813
NIC Algorithm	0.9162	0.9977	0.0023	0.0838	0.5142	0.9372	0.9266
overpass							
Average Filtering	0.6961	0.9374	0.0626	0.3039	6.5856	0.1312	0.2207
Median Filtering	0.6976	0.9526	0.0474	0.3024	5.0800	0.1666	0.2690
Average Running	0.6384	0.9793	0.0207	0.3616	2.5307	0.2948	0.4034
Approximated Median	0.6629	0.9844	0.0156	0.3371	1.9942	0.3654	0.4711
Running Gassian Average	0.5367	0.9732	0.0268	0.4633	3.2684	0.2136	0.3056
Histogram Over Time	0.7235	0.9286	0.0714	0.2765	7.4113	0.1210	0.2074
NIC Algorithm	0.8091	0.9996	0.0004	0.1909	0.2941	0.9658	0.8805
badminton							
Average Filtering	0.7270	0.9594	0.0406	0.2730	4.8581	0.3886	0.5065
Median Filtering	0.7151	0.9686	0.0314	0.2849	4.0058	0.4474	0.5504
Average Running	0.6205	0.9549	0.0451	0.3795	5.6572	0.3281	0.4293
Approximated Median	0.6797	0.9646	0.0354	0.3203	4.5160	0.4054	0.5079
Running Gassian Average	0.4934	0.9885	0.0115	0.5066	2.8466	0.6039	0.5431
Histogram Over Time	0.6806	0.9186	0.0814	0.3194	8.9552	0.2289	0.3426
NIC Algorithm	0.7777	0.9909	0.0091	0.2223	1.6414	0.7520	0.7647

Table 4.8: Continued

Method	Re	Sp	FPR	FNR	PWC	Pre	F1
traffic							
Average Filtering	0.7233	0.9223	0.0777	0.2767	9.0095	0.3819	0.4999
Median Filtering	0.7400	0.9335	0.0665	0.2600	7.8543	0.4249	0.5398
Average Running	0.6192	0.8882	0.1118	0.3808	12.8565	0.2688	0.3748
Approximated Median	0.7453	0.9398	0.0602	0.2547	7.2312	0.4511	0.5620
Running Gassian Average	0.6572	0.9647	0.0353	0.3428	5.4448	0.5527	0.6004
Histogram Over Time	0.7393	0.9187	0.0813	0.2607	9.2483	0.3764	0.4988
NIC Algorithm	0.7492	0.9860	0.0140	0.2508	2.8775	0.7799	0.7642
sofa							
Average Filtering	0.3374	0.9954	0.0046	0.6626	3.3308	0.7710	0.4694
Median Filtering	0.3317	0.9968	0.0032	0.6683	3.2201	0.8274	0.4736
Average Running	0.3100	0.9970	0.0030	0.6900	3.3024	0.8238	0.4505
Approximated Median	0.2489	0.9987	0.0013	0.7511	3.4013	0.8992	0.3899
Running Gassian Average	0.7543	0.9566	0.0434	0.2457	5.2257	0.4423	0.5576
Histogram Over Time	0.4294	0.9998	0.0002	0.5706	2.5108	0.9898	0.5990
NIC Algorithm	0.6537	0.9809	0.0191	0.3463	3.3423	0.6093	0.6307
parking							
Average Filtering	0.7156	0.9615	0.0385	0.2844	5.7564	0.6087	0.6578
Median Filtering	0.7299	0.9732	0.0268	0.2701	4.5629	0.6953	0.7121
Average Running	0.2386	0.9936	0.0064	0.7614	6.4774	0.7580	0.3629
Approximated Median	0.1448	0.9969	0.0031	0.8552	6.9002	0.7962	0.2450
Running Gassian Average	0.5102	0.9336	0.0664	0.4898	9.9140	0.3917	0.4432
Histogram Over Time	0.9534	0.7680	0.2320	0.0466	21.7655	0.2562	0.4039
NIC Algorithm	0.7708	0.9477	0.0523	0.2292	6.6000	0.5525	0.6436
skating							
Average Filtering	0.7600	0.9195	0.0805	0.2400	8.8356	0.3296	0.4598
Median Filtering	0.7636	0.9969	0.0031	0.2364	1.4643	0.9277	0.8377
Average Running	0.6476	0.9509	0.0491	0.3524	6.4079	0.4073	0.5000
Approximated Median	0.7501	0.9969	0.0031	0.2499	1.5333	0.9260	0.8288
Running Gassian Average	0.7032	0.9954	0.0046	0.2968	1.9086	0.8878	0.7848
Histogram Over Time	0.7329	0.9778	0.0222	0.2671	3.4366	0.6317	0.6785
NIC Algorithm	0.9187	0.9944	0.0056	0.0813	0.9323	0.8956	0.9070
blizzard							
Average Filtering	0.6859	0.9997	0.0003	0.3141	0.3942	0.9661	0.8023
Median Filtering	0.7192	0.9995	0.0005	0.2808	0.3802	0.9407	0.8152
Average Running	0.6522	0.9999	0.0001	0.3478	0.4172	0.9849	0.7847
Approximated Median	0.5706	0.9999	0.0001	0.4294	0.5103	0.9858	0.7228
Running Gassian Average	0.8262	0.9962	0.0038	0.1738	0.5783	0.7194	0.7691
Histogram Over Time	0.7235	0.9990	0.0010	0.2765	0.4251	0.8915	0.7987
NIC Algorithm	0.8024	0.9994	0.0006	0.1976	0.2909	0.9392	0.8654



(a)



(b)

Figure 4.8: Continued: (a) *skating*, and (b) *blizzard*.

rithm significantly improves the detection accuracy. In Table 4.10, we show the processing speed results of all testing methods in FPS metric. It is observed that most of the testing methods based on updating the background reference image yields impressive processing speed results. By reducing the parameter of buffer sampling rate, filtering approaches and histogram over time method consume less computational cost for updating the buffer and also estimating the background image. Although the updating progress is executed for each input frame in average running, approximated median, and running Gaussian average, their updating schemes are quite simple to reasonably re-

Table 4.9: Average accuracy comparison of NIC to several baseline methods of background reference image based.

Method	Re	Sp	FPR	FNR	PWC	Pre	F1
Average Filtering	0.6954	0.9374	0.0626	0.3046	7.3995	0.5274	0.5285
Median Filtering	0.6861	0.9566	0.0434	0.3139	5.6365	0.6230	0.5846
Average Running	0.5906	0.9611	0.0389	0.4094	5.7339	0.5792	0.5099
Approximated Median	0.5673	0.9670	0.0330	0.4327	5.3217	0.6567	0.5227
Running Gassian Average	0.6320	0.9629	0.0371	0.3680	5.1552	0.5238	0.5502
Histogram Over Time	0.7502	0.9190	0.0810	0.2498	8.7575	0.5436	0.5404
NIC Algorithm	**0.8201**	**0.9884**	**0.0116**	**0.1799**	**1.8979**	**0.8245**	**0.8191**

Table 4.10: Average processing speed (FPS) comparison

Method	Processing Speed (FPS)
Average Filtering	∼31.0 FPS
Median Filtering	∼18.9 FPS
Average Running	∼18.6 FPS
Approximated Median	∼29.5 FPS
Running Gaussian Average	∼20.9 FPS
Histogram Over Time	∼12.0 FPS
NIC Algorithm	∼9.0 FPS

Table 4.11: Parameter configuration of comparing methods

Method	Parameter Configuration
KDE [48]	$N = 100$
EGMM [40]	$K = 3$, $\alpha = 0.001$
ViBE [60]	$N = 20$, $R = 20$, $\phi = 16$, $\#_{\min} = 3$
PBAS [64]	$N = 35$, $\#_{\min} = 3$, $R_{\frac{inc}{dec}} = 0.05$, $R_{lower} = 18$, $R_{scale} = 5$, $T_{dec} = 0.05$, $T_{inc} = 1$, $T_{lower} = 2$, $T_{upper} = 100$, $\alpha = 10$, MedFilter 9×9
Simp-SOBS [74]	$\alpha_1 = 0.02$, $\alpha_1 = 0.01$
NIC	$\tau = 20$, Mask Size 7×7

port high values for FPS. Besides, another advantage, if compared with filtering approaches and histogram over time, is that no memory is required for the buffer. Compared with traditional background subtraction algorithms based estimating the background reference image, NIC algorithm achieves the remarkable accuracy with the comparative processing speed. Our proposed algorithm is more suitable than other methods in highly challenging background scenarios while adapts a high-speed processing for low-cost computation systems.

4.3.2 Comparing with Background Modeling Methods

We compare the proposed method with five well-known background subtraction methods in the group of statistical model: the Kernel Density Estimation (KDE) in [48], the improved adaptive GMM (EGMM) in [40], the Visual Background Extractor (ViBE) in [60], the Pixel-based Adaptive Segmenter (PBAS) in [64], and the Simplified Self-Organized Background Subtraction (Simp-SOBS) in [74]. The parameter configuration of all above methods are summarized in Table 4.11. All experiment results of other methods used for comparison and reported in this section are obtained from their outcomes, only the NIC results are done by own implementation. It is noted that the results of PBAS for *skating* and *blizzard* are unavailable.

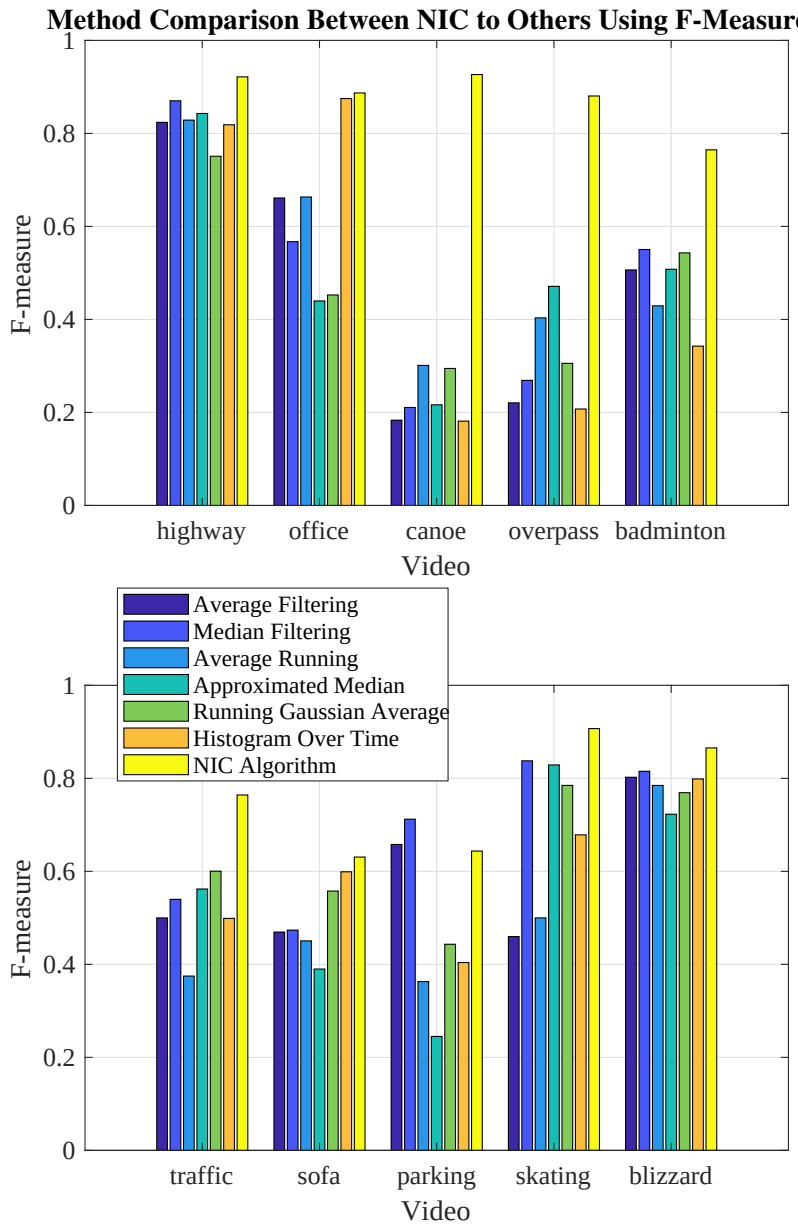


Figure 4.9: F-measure comparison of NIC and several baseline methods of background reference image on particular videos.

Table 4.12: Foreground detection accuracy comparison of NIC with several state-of-the-art methods of background model

Method	Re	Sp	FPR	FNR	PWC	Pre	F1
highway							
KDE	0.9379	0.9957	0.0043	0.0621	0.7686	0.9328	0.9353
EGMM	0.8916	0.9949	0.0051	0.1084	1.1250	0.9163	0.9038
ViBE	0.7475	0.9998	0.0002	0.2525	2.2289	0.9975	0.8546
PBAS	0.9587	0.9956	0.0044	0.0413	0.6602	0.9318	0.9451
Simp-SOBS	0.5573	0.9954	0.0046	0.4427	3.0535	0.8847	0.6838
NIC	0.9140	0.9956	0.0044	0.0860	0.9219	0.9292	0.9216
office							
KDE	0.9055	0.9977	0.0023	0.0945	0.8622	0.9676	0.9355
EGMM	0.5075	0.9971	0.0029	0.4925	3.6679	0.9290	0.6564
ViBE	0.7023	0.9983	0.0017	0.2977	2.8400	0.9765	0.8170
PBAS	0.9292	0.9968	0.0032	0.0708	0.7906	0.9551	0.9420
Simp-SOBS	0.1759	0.9914	0.0086	0.8241	6.4863	0.6038	0.2725
NIC	0.8894	0.9914	0.0086	0.1106	1.5639	0.8847	0.8870
canoe							
KDE	0.8315	0.9980	0.0020	0.1685	0.7860	0.9396	0.8822
EGMM	0.8533	0.9973	0.0027	0.1467	0.7844	0.9194	0.8851
ViBE	0.6681	0.9793	0.0207	0.3319	3.5868	0.6220	0.6442
PBAS	0.5625	1.0000	0.0000	0.4375	1.5522	0.9986	0.7196
Simp-SOBS	0.4333	0.9473	0.0527	0.5667	5.9630	0.1004	0.1630
NIC	0.9162	0.9977	0.0023	0.0838	0.5142	0.9372	0.9266
overpass							
KDE	0.8003	0.9981	0.0019	0.1997	0.4550	0.8512	0.8250
EGMM	0.8076	0.9993	0.0007	0.1924	0.3311	0.9366	0.8673
ViBE	0.7905	0.9655	0.0345	0.2095	5.0200	0.6928	0.7384
PBAS	0.6704	0.9997	0.0003	0.3296	0.4703	0.9690	0.7925
Simp-SOBS	0.2240	0.8605	0.1395	0.7760	14.3509	0.0100	0.0192
NIC	0.8091	0.9996	0.0004	0.1909	0.2941	0.9658	0.8805
badminton							
KDE	0.7904	0.9860	0.0140	0.2096	2.0732	0.6667	0.7233
EGMM	0.7147	0.9848	0.0152	0.2853	2.4480	0.6251	0.6669
ViBE	0.7015	0.9946	0.0054	0.2985	2.0548	0.8767	0.7794
PBAS	0.7996	0.9972	0.0028	0.2004	0.9549	0.9110	0.8517
Simp-SOBS	0.6937	0.9466	0.0534	0.3063	6.9180	0.4629	0.5552
NIC	0.7777	0.9909	0.0091	0.2223	1.6414	0.7520	0.7647

Table 4.13: Continued

Method	Re	Sp	FPR	FNR	PWC	Pre	F1
traffic							
KDE	0.8589	0.9284	0.0716	0.1411	7.5974	0.4431	0.5846
EGMM	0.7368	0.9559	0.0441	0.2632	5.7746	0.5258	0.6137
ViBE	0.6460	0.9984	0.0016	0.3540	0.8349	0.8836	0.7464
PBAS	0.8497	0.9721	0.0279	0.1503	3.5549	0.6688	0.7485
Simp-SOBS	0.6110	0.9782	0.0218	0.3890	3.4357	0.4992	0.5495
NIC	0.7492	0.9860	0.0140	0.2508	2.8775	0.7799	0.7642
sofa							
KDE	0.5191	0.9961	0.0039	0.4809	2.4774	0.8572	0.6466
EGMM	0.5141	0.9972	0.0028	0.4859	2.3919	0.8925	0.6524
ViBE	0.3847	0.9983	0.0017	0.6153	4.1428	0.9405	0.5461
PBAS	0.6180	0.9974	0.0026	0.3820	1.9153	0.9160	0.7381
Simp-SOBS	0.2055	0.9985	0.0015	0.7945	3.6100	0.8644	0.3320
NIC	0.6537	0.9809	0.0191	0.3463	3.3423	0.6093	0.6307
parking							
KDE	0.5191	0.9961	0.0039	0.4809	2.4774	0.8572	0.6466
EGMM	0.5141	0.9972	0.0028	0.4859	2.3919	0.8925	0.6524
ViBE	0.3847	0.9983	0.0017	0.6153	4.1428	0.9405	0.5461
BPAS	0.0958	0.9993	0.0007	0.9042	7.0551	0.9220	0.1736
Simp-SOBS	0.2055	0.9985	0.0015	0.7945	3.6100	0.8644	0.3320
NIC	0.7708	0.9477	0.0523	0.2292	6.6000	0.5525	0.6436
skating							
KDE	0.8699	0.9981	0.0019	0.1301	0.6992	0.9499	0.9081
EGMM	0.7811	0.9989	0.0011	0.2189	0.9736	0.9676	0.8644
ViBE	0.8470	0.9774	0.0226	0.1530	2.8181	0.6238	0.7185
PBAS	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Simp-SOBS	0.7049	0.9947	0.0053	0.2951	1.6860	0.8450	0.7686
NIC	0.9187	0.9944	0.0056	0.0813	0.9323	0.8956	0.9070
blizzard							
KDE	0.6331	0.9999	0.0001	0.3669	0.4311	0.9890	0.7720
EGMM	0.7849	0.9967	0.0033	0.2151	0.5762	0.7338	0.7585
ViBE	0.6366	0.9997	0.0003	0.3634	0.3635	0.9476	0.7616
PBAS	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Simp-SOBS	0.6453	0.9987	0.0013	0.3547	0.5370	0.8531	0.7348
NIC	0.8024	0.9994	0.0006	0.1976	0.2909	0.9392	0.8654

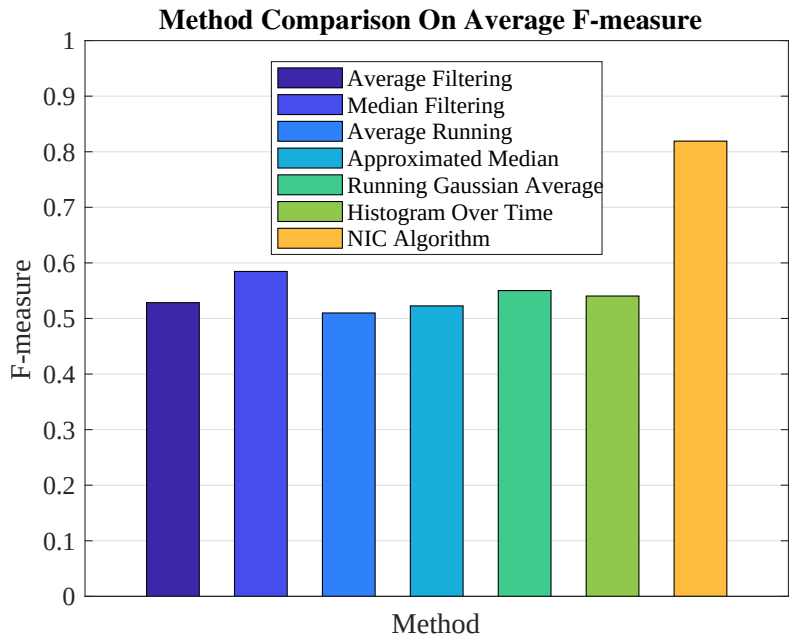
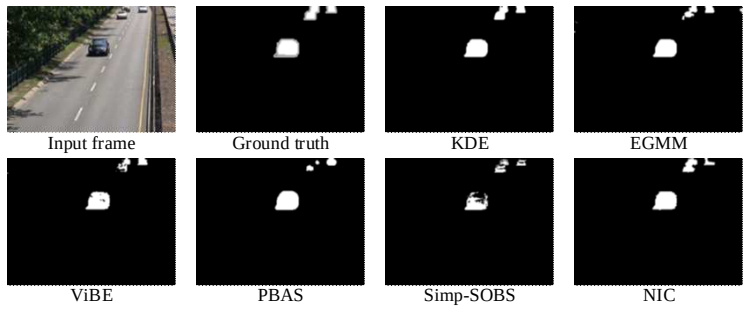


Figure 4.10: Average F-measure comparison of NIC and several baseline methods of background reference image based.

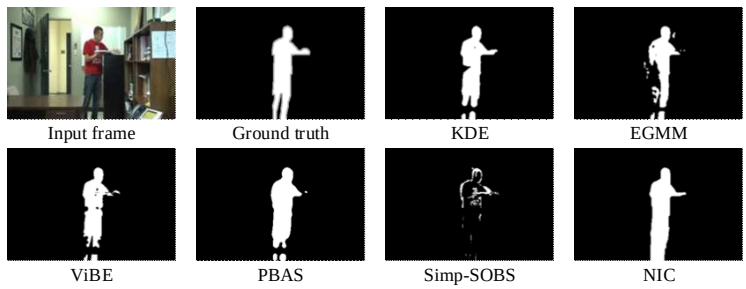
At first, we present the qualitative comparison of all methods in Fig. 4.11, 4.12, and 4.13. Compared to others, Simp-SOBS usually gives the worst results of foreground detection, e.g. much noise from background motion appears in *canoe*, *overpass*, *traffic*. In *office*, Simp-SOBS mostly fails to detect the object as Figure 4.11(b). As the predicted results, its FPR value is high due to background noise confusion, and its Re, Sp, PWC, Pre, F1 values are low due to the failure of detection. For remaining methods, the visual results are relatively similar in qualitative assessment, that means, it is not easy to recognize which ones are better. However, it can be observed noises

Table 4.14: Average accuracy comparison of NIC to several state-of-the-art methods of background model based.

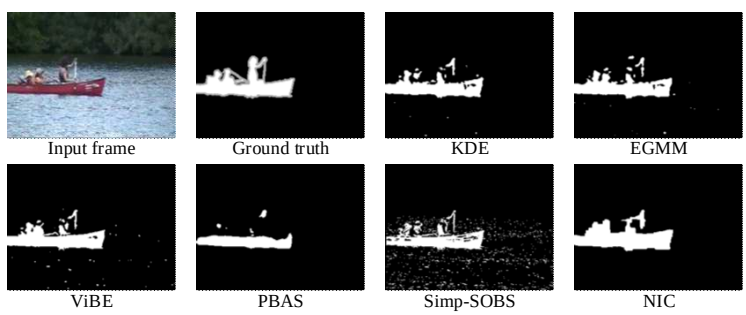
Method	Re	Sp	FPR	FNR	PWC	Pre	F1
KDE	0.7414	0.9884	0.0116	0.2586	2.3107	0.8213	0.7586
EGMM	0.7290	0.9901	0.0099	0.2710	2.2319	0.8184	0.7586
ViBE	0.6372	0.9907	0.0093	0.3628	3.4633	0.8455	0.6994
PBAS	0.6855	**0.9948**	**0.0052**	0.3145	2.1192	**0.9090**	0.7389
Simp-SOBS	0.4412	0.9705	0.0295	0.5588	5.3149	0.5791	0.4338
NIC	**0.8201**	0.9884	0.0116	**0.1799**	**1.8979**	0.8245	**0.8191**



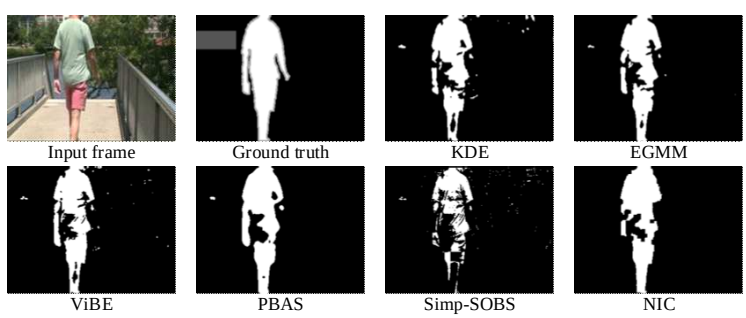
(a)



(b)

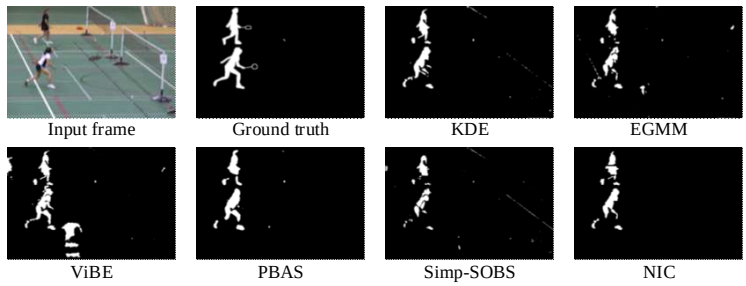


(c)

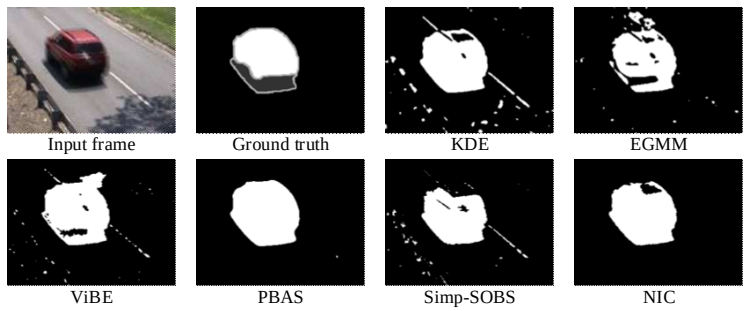


(d)

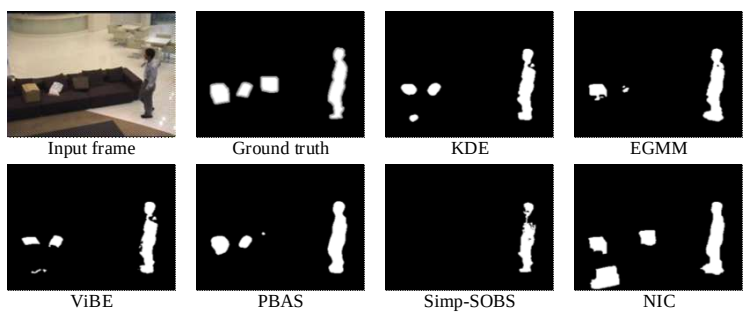
Figure 4.11: Foreground images of NIC and other model based background subtraction methods: (a) *highway*, (b) *office*, (c) *canoe*, and (d) *overpass*.



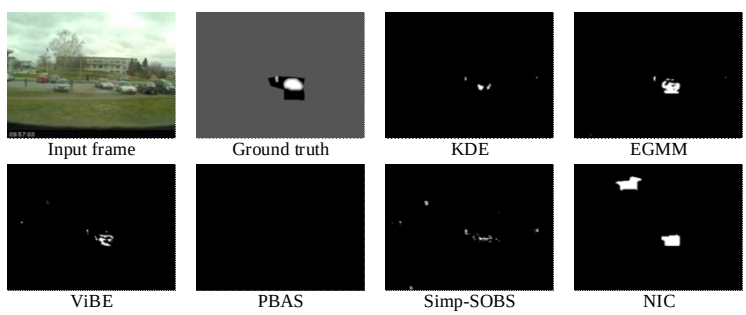
(a)



(b)

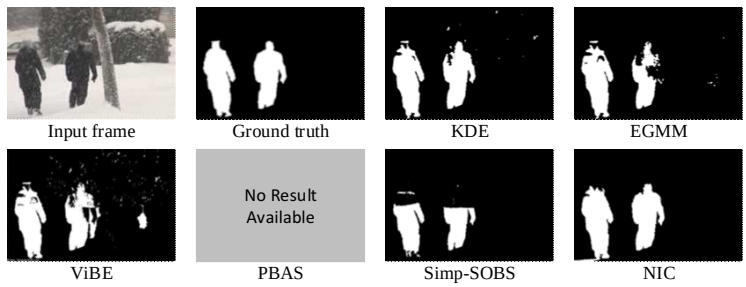


(c)

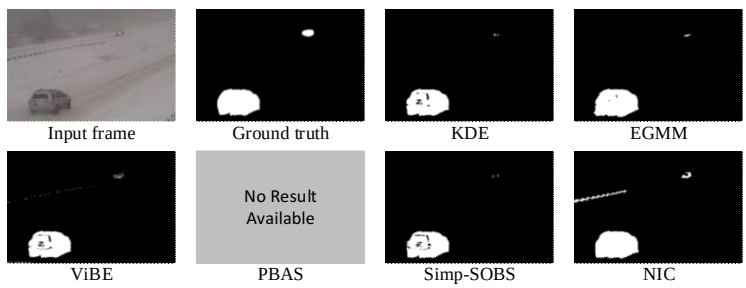


(d)

Figure 4.12: Continued: (a) *badminton*, (b) *traffic*, (c) *sofa*, and (d) *parking*.



(a)



(b)

Figure 4.13: Continued: (a) *skating*, and (b) *blizzard*.

from background motion and camera vibration appear in the results of KDE, EGMM, and ViBE. Moreover, most of the compared methods cannot detect object areas which are homogenous with the background, e.g. a box in *sofa* and human legs in *office*. A noticeable thing is that most methods are unable to detect and further eliminate object shadows, except KDE, but it seems inaccurate as the result in Figure 4.12(b).

The quantitative comparison of NIC and other state-of-the-art methods is reported in Table 4.12, 4.13, 4.14, and Figure 4.14, 4.15. Simp-SOBS, a background subtraction method developed from self-organized mapping architecture, produces worst results at all quantitative evaluations. Although Simp-SOBS is developed to adapt to background variations like illumination and slow moving background, it is so fragile to dynamic background motion as *canoe*, *overpass*, intermittent object motion as *office*, *sofa*, *parking*, and camera jitter as *traffic*, *badminton*. KDE utilizes several input frames to construct a background model and updates this model directly based on the intensity density from sample history values. Since KDE updates the background model sequentially, it is able to handle quickly changing illuminance and also small background motion.

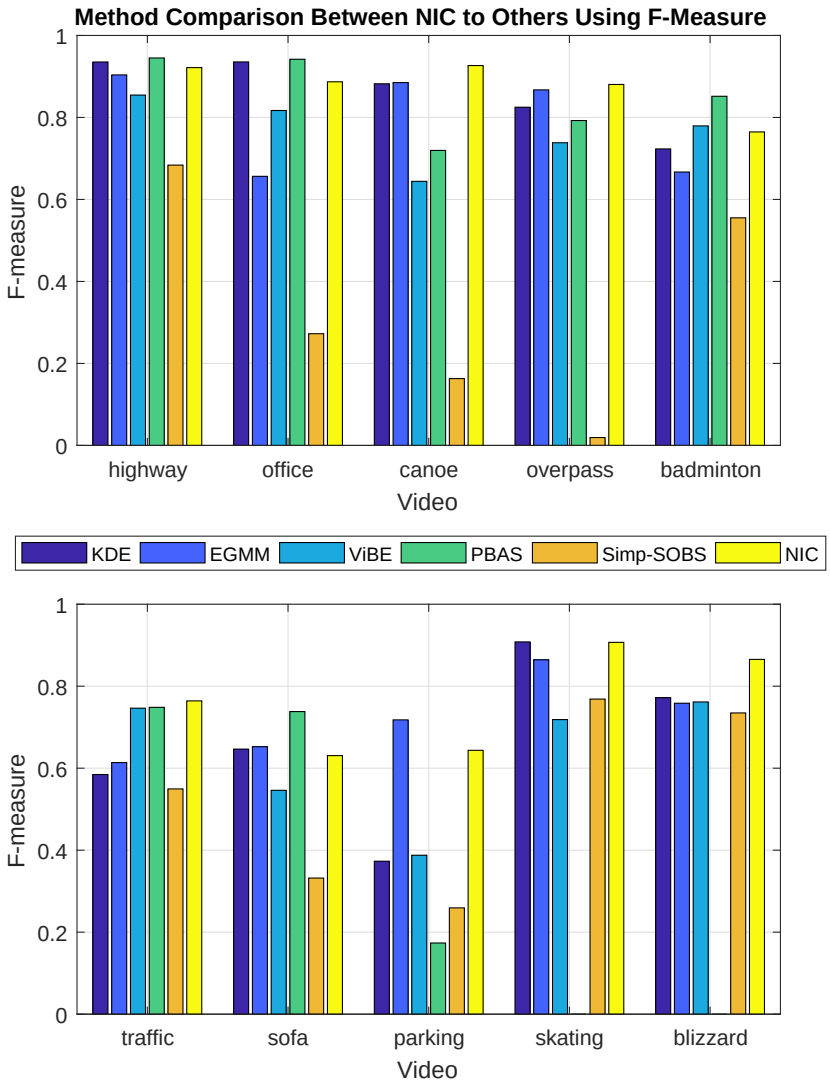


Figure 4.14: F-measure comparison of NIC and several state-of-the-art methods of background model on particular videos .

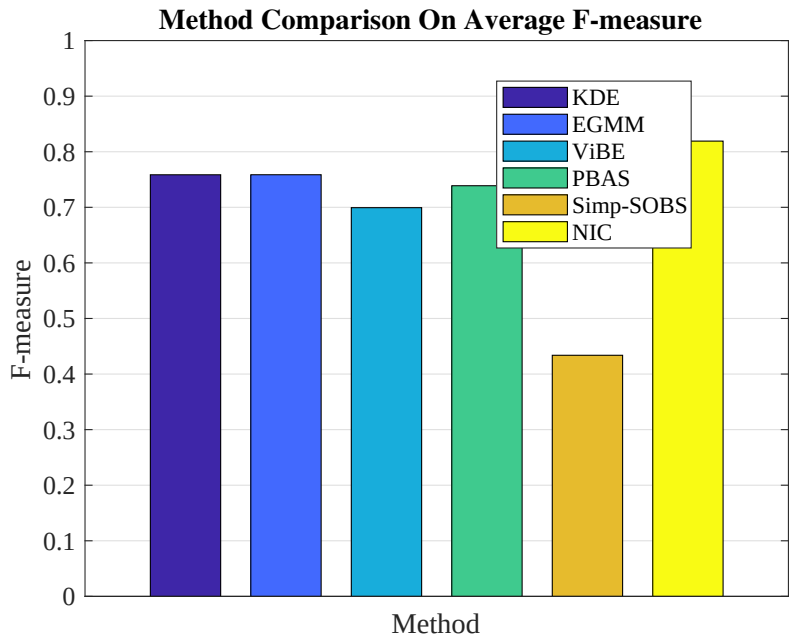


Figure 4.15: Average F-measure comparison of NIC and several state-of-the-art methods of background model based.

The EGMM method updates the parameters and selects the appropriate number of Gaussian components by recursive equations simultaneously. Compared with the original GMM [37], EGMM achieves a remarkable improvement of processing speed. Based on the quantitative results, both KDE and EGMM are fairly efficient to middle mixture challenge as *highway*, dynamic background challenge as *canoe*, *overpass*, and bad weather with poor recording condition as *skating*, *blizzard*. A combination of a short-term and a long-term patterns in KDE allows adapting to very slow changes and be more proficient than EGMM in a situation where objects staying for a while. However, they cannot deal with the more challenging samples in the category of intermittent object motion and camera jitter. It can be seen that almost methods yield poor detection accuracy of *sofa* and *parking*, except PBAS. ViBE, an approach updates its background model over time by a lifespan policy, reports unremarkable outcomes of foreground detection if compared with PBAS and NIC. However, ViBE outperforms KDE and EGMM in the camera vibration challenge. It is realized that that the strength of ViBE is very low computational cost. By an efficient scheme which allows updating the background model with pixel-wise learning rate, PBAS presents a sta-

Table 4.15: Processing speed comparison between NIC and several state-of-the-art methods of background model based.

Method	Facility	Processing Speed (FPS)
KDE	C++ on Core i7 3.4GHz	$\sim$9 FPS on 720×480 video
EGMM	C++ on Core i7 3.4GHz	$\sim$49 FPS on 720×480 video
ViBE	C on Core i7 2.67GHz	$\sim$180 FPS on 640×480 video
PBAS	C++ on Core i7 3.5GHz	$\sim$48 FPS (average of different resolutions)
Simp-SOBS	Matlab on Core i7 2.3GHz laptop	$\sim$0.06 FPS on 720×576 video
NIC	Matlab on Core i7 2.67GHz laptop	$\sim$**9** FPS on 720×480 video

ble result for various background challenges. For instance, PBAS is the winner of *highway*, *office* in the baseline, *badminton* in the camera jitter, and *sofa* in the intermittent object motion, however, it delivers the worst F-measure of *parking*. From Table 4.14, PBAS realizes the best score of Sp, FPR, and Pre. Similar to ViBE, PBAS is more sensitive to background motion. The average results reported in Table 4.14 and shown in Figure 4.15 do not include *skating* and *blizzard*. Compared to the existing state-of-the-art approaches, NIC achieves the comparative results for all testing videos. In particular, NIC takes the best scores of four over seven benchmark metrics such as Re, FNR, PWC, and F1. NIC is especially good at dynamic background challenge. If compared to the runner-up of overall F-measure, NIC improves approximately $\sim 8\%$. As another benefit, NIC requires fewer parameters for tuning, e.g. only 2 (constant threshold (tau and mask size) vs 4 of EGMM and 11 of PBAS.

In the last experiment, we compare the computational complexity based on processing speed measured by fps metric. The information relating to facility used for evaluation is described in Table 4.15. It is realized that the advantage of ViBE is high-speed processing. Besides the standard version, there is a down-scale version of ViBE for device implementation. Estimating the background probabilities at every pixel in KDE is a cause of slower processing speed if compared with EGMM. In this experiment, PBAS reports the average fps of different resolutions including 320×240, 360×240, 540×360, 720×480. It should be noted that KDE, EGMM, ViBE, and PBAS are evaluated on C or C++ platform while Simp-SOBS and NIC are benchmarked on Matlab environment by laptop. Although there is a minor difference in the used facility (2.67GHz vs 2.3GHz), it is not difficult to recognize a big margin in fps results of NIC and Simp-SOBS, e.g. $\sim$9fps of NIC vs $\sim$0.06fps of Simp-SOBS. Utilization of self-organizing map, a type of ar-

tificial neural network for unsupervised learning, is the major reason of low processing speed of Simp-SOBS.

Chapter 5

Conclusion and Future Direction

5.1 Conclusion

In the field of image processing and computer vision, foreground detection, known as background subtraction, is currently an opening research due to its important role in many multimedia applications. Over the last decade, there are so many background subtraction methods introduced for accurately detecting foreground image. Although background reference image based approaches are quite simple and fast processing, they can not deal with various background challenges. The detection accuracy is significantly improved by background model based approaches which are more complex and time-consuming. Obviously, most of modern background subtraction approaches are unsuitable to implement in low-price systems such as video surveillance system based on IP cameras which are specified by low computational cost and less memory capacity. Developing a novel method that is expected to efficiently work with various background challenges while maintaining a high-speed processing is the real challenge. According to above motivations, we have proposed an efficient background estimation algorithm, namely Neighbor-based Intensity Correction (NIC), which allows modeling the background as a reference image.

- A novel background maintenance scheme allows to directly update the background reference image for each input frame. The scheme has an ability to correct the intensity of object pixels by the intensity of background pixels by a proposed intensity updating rule. Based on analyzing the homogeneity of intensity patterns captured in the current background image and the input frame, pre-identified motion pixels are updated to a higher homogeneous pattern. In order to calculate the pattern homogeneity, the standard deviation metric is used, e.g. a smaller standard deviation representing a higher homogeneous pattern. Thus, the de-

cision of intensity correction is made by updating rule. Since the pattern acquires not only the object pixel but also the neighboring pixels, quality of updating progress is generally improved.

- A novel factor, namely steadiness, which is capable of measuring the variation of pixel intensity between the background image and the input frame. This factor is calculated for all pixels based on the result of difference extraction and then maintained in a steadiness matrix. Detection and elimination of infrequent background motion pixels as noise pixels by this factor allows reducing the computation cost of the updating process. Obviously, besides maintaining the background estimation accuracy, the processing speed is much improved by the selective updating scheme based on measuring steadiness.

- The proposed NIC algorithm for foreground detection is evaluated on various video samples representing different background challenges. The algorithm is investigated under various parameter configurations to analyze the performance in the terms of foreground detection accuracy and processing speed. As the result of CDNET 2014, NIC algorithm achieves 0.8191 of F-measure metric of foreground detection and $\sim$9fps of processing speed on 720×480 video. If compared with traditional methods which model background as a reference image, NIC significantly improves the accuracy of foreground detection with the comparable result of FPS. If compared with five recently well-know background subtraction methods working on background models, NIC improves at least $\sim 8\%$ approximately of overall F-measure. Due to modeling the background as a reference image, the proposed NIC algorithm has some highlight features as low computational cost, less memory consumption besides accurate foreground detection from the efficient background updating scheme.

5.2 Future Direction

As analyzed and discussed in the experiment section, the drawback of NIC algorithm is collecting objects into the background class in the case of detected objects staying in a scene for a long time. This behavior detects inaccurate foreground, e.g., (i) objects cannot be detected during non-moving duration due to its being in a scene, and (ii) objects are detected in a scene even if they

already moved out. Although NIC algorithm can overcome this problem, it will take much time, e.g., NIC requires several input frames to update background image again. It is realized that the response of NIC for this situation is not sensitive enough. Therefore, besides improving the background estimation accuracy, a novel responding scheme should be developed and integrated into NIC to efficiently handle the intermittent object motion challenge. In particular, there are two improvements which can be directly applied to NIC

- Currently, NIC uses a fixed square mask to capture the intensity pattern of neighbor pixels surrounding a motion pixel. Obviously, only one mask for various kinds of movements including velocity and direction is not an optimal solution. Thus, an automatic mask selection scheme which is able to select the most appropriate binary mask for a particular object moving style is necessary.

- To effectively deal with the realistic challenges, we realize that only standard deviation used to measure the pattern homogeneity is not robust. In the case where the object area is more homogeneous in the intensity than the complex background area at the local level, NIC algorithm will be failed. A potential solution is combining standard deviation with other metrics to analyze pattern homogeneity thoughtfully and accurately.

Besides moving object detection and tracking, the proposed background subtraction is potential to apply for human pose estimation and action recognition. Several human pose estimation methods require the foreground mask of pixel-wise human segmentation as the input [83–88]. Some challenges in the field of human pose estimation, especially with unconstrained videos, relates to the variation of observation and inference. Most of the existing methods currently solve the problem in stationary camera environment. Some of them provides a solution for moving camera, however, the performance is not impressive with low accuracy and high computational cost. In general framework of human pose estimation, noisy silhouettes extracted by background subtraction might affect the accuracy of pose estimation. Since the human pose estimation is mostly applied for indoor applications, in which human is detected in a pleasing background without illumination change and dynamic background motion, the challenge of background modeling is insignificant or even ignored for consideration. Outcomes of human pose estimation are usually developed for action recognition in surveillance systems and interactive gaming systems.

Background subtraction method is further used for automatic human detection and action recognition. In the general framework of human action recognition, foreground detection contributes as a pre-processing step. Similar to human pose estimation, the main challenges in the field of action recognition consists of variety of realistic actions and human representation. Based on the foreground mask, the area of human in a scene is extracted for sequential processes as feature extraction and action model training. Recognizing basic actions [92–95] likes walking, running, sitting, jumping, waving hands in an image sequence is considered in computer vision systems as human-computer interaction, video content retrieval and surveillance. Due to concentrating on the problem of the variety of object appearance, the role of background subtraction in the action recognition framework is mostly ignored or marginally discussed. Some of them usually assume that the human appearance in a scene is detected by a background subtraction method. Most of the current datasets of action recognition are recorded in pleasing environments without much more challenging background scenarios.

Bibliography

[1] C. G. del Postigo, J. Torres, and J. M. Menéndez, "Vacant parking area estimation through background subtraction and transience map analysis," *IET Intelligent Transport Systems*, vol. 9, no. 9, pp. 835–841, 2015.

[2] L. Unzueta, M. Nieto, A. Cortes, J. Barandiaran, O. Otaegui, and P. Sanchez, "Adaptive multicue background subtraction for robust vehicle counting and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 527–540, June 2012.

[3] E. Cermeño, A. Pérez, and J. A. Sigüenza, "Intelligent video surveillance beyond robust background modeling," *Expert Systems with Applications*, vol. 91, pp. 138 – 149, 2018.

[4] I. Kim, T. W. Awan, and Y. Soh, "Background subtraction-based multiple object tracking using particle filter," in *IWSSIP 2014 Proceedings*, May 2014, pp. 71–74.

[5] K. C. Chang, P. K. Liu, and Y. S. Wang, "Parallel design of background subtraction and template matching modules for image objects tracking system," in *2016 International Computer Symposium (ICS)*, Dec 2016, pp. 18–21.

[6] E. Surendar, V. M. Thomas, and A. M. Posonia, "Animal tracking using background subtraction on multi threshold segmentation," in *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, March 2016, pp. 1–6.

[7] N. Sirikuntamat, S. Satoh, and T. H. Chalidabhongse, "Vehicle tracking in low hue contrast based on camshift and background subtraction," in *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, July 2015, pp. 58–62.

[8] J. H. Ahn, "Fast adaptive robust subspace tracking for online background subtraction," in *2014 22nd International Conference on Pattern Recognition*, Aug 2014, pp. 2555–2559.

[9] Y. Tian, R. S. Feris, H. Liu, A. Hampapur, and M. T. Sun, "Robust detection of abandoned and removed objects in complex surveillance videos," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 5, pp. 565–576, Sept 2011.

[10] D. J. Jayamanne, J. Samarawickrama, and R. Rodrigo, "Appearance based tracking with background subtraction," in *2013 8th International Conference on Computer Science Education*, April 2013, pp. 643–649.

[11] D. D. Bloisi, A. Pennisi, and L. Iocchi, "Background modeling in the maritime domain," *Machine Vision and Applications*, vol. 25, no. 5, pp. 1257–1269, Jul 2014.

[12] A. T. Nghiem and F. Bremond, "Background subtraction in people detection framework for rgb-d cameras," in *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2014, pp. 241–246.

[13] F. Xu and K. Fujimura, "Human detection using depth and gray images," in *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.*, July 2003, pp. 115–121.

[14] D. D. Doyle, A. L. Jennings, and J. T. Black, "Optical flow background subtraction for real-time ptz camera object tracking," in *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, May 2013, pp. 866–871.

[15] B. Lee and M. Hedley, "Background estimation for video surveillance," in *Proceedings. Image & Vision Computing New Zealand (IVCNZ '02)*, 2002, pp. 315–320.

[16] H. Zhiwei, L. Jilin, and L. Peihong, "New method of background update for video-based vehicle detection," in *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems*, Oct 2004, pp. 580–584.

[17] W.-x. Kang, W.-z. Lai, and X.-b. Meng, "An adaptive background reconstruction algorithm based on inertial filtering," *Optoelectronics Letters*, vol. 5, no. 6, pp. 468–471, Nov 2009.

[18] F.-C. Cheng, S.-C. Huang, and S.-J. Ruan, "Advanced motion detection for intelligent video surveillance systems," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, ser. SAC '10. New York, NY, USA: ACM, 2010, pp. 983–984.

[19] N. J. B. McFarlane and C. P. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications*, vol. 8, no. 3, pp. 187–193, May 1995.

[20] P. Shi, E. G. Jones, and Q. Zhu, "Median model for background subtraction in intelligent transportation system," in *Image Processing: Algorithms and Systems III*, vol. 5298, May 2004, pp. 168–176.

[21] H. Al-Khateeb and M. Petrou, "Automatic change detection in an indoor environment," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, June 2010, pp. 53–58.

[22] M. H. Hung, J. S. Pan, and C. H. Hsieh, "Speed up temporal median filter for background subtraction," in *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, Sept 2010, pp. 297–300.

[23] N. N. E. H. J. Zheng, Y. Wang, "Extracting roadway background image: A mode based approach," *Transportation Research Board*, 2006.

[24] A.-N. Lai, H. Yoon, and G. Lee, "Robust background extraction scheme using histogram-wise for real-time tracking in urban traffic video," in *2008 8th IEEE International Conference on Computer and Information Technology*, July 2008, pp. 845–850.

[25] S. Zhang, H. Yao, and S. Liu, "Dynamic background subtraction based on local dependency histogram," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 07, pp. 1397–1419, 2009.

[26] B. Zhong, H. Yao, S. Liu, and X. Yuan, "Local histogram of figure/ground segmentations for dynamic background subtraction," *EURASIP J. Adv. Signal Process*, vol. 2010, pp. 55:1–55:14, Feb. 2010.

[27] Y. Hoshen, C. Arora, Y. Poleg, and S. Peleg, "Efficient representation of distributions for background subtraction," in *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, Aug 2013, pp. 276–281.

[28] S. M. Roy and A. Ghosh, "Real-time adaptive histogram min-max bucket (hmmb) model for background subtraction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[29] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, Jul 1997.

[30] M. A. Rahman, B. Ahmed, M. A. Hossian, and M. N. I. Mondal, "An adaptive background modeling based on modified running gaussian average method," in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Feb 2017, pp. 524–527.

[31] J. Pan, C.-W. Lin, C. Gu, and M.-T. Sun, "A robust video object segmentation scheme with prestored background information," in *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, vol. 3, 2002, pp. 803–806.

[32] T. Yang, S. Z. Li, Q. Pan, and J. Li, "Real-time and accurate segmentation of moving objects in dynamic scene," in *Proceedings of the ACM 2Nd International Workshop on Video Surveillance &Amp; Sensor Networks*, ser. VSSN '04. New York, NY, USA: ACM, 2004, pp. 136–143.

[33] J. Lluís, X. Miralles, and O. Bastidas, "Reliable real-time foreground detection for video surveillance applications," in *Proceedings of the Third ACM International Workshop on Video Surveillance &Amp; Sensor Networks*, ser. VSSN '05. New York, NY, USA: ACM, 2005, pp. 59–62.

[34] M. F. Abdelkader, R. Chellappa, Q. Zheng, and A. L. Chan, "Integrated motion detection and tracking for visual surveillance," in *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, Jan 2006, pp. 28–28.

[35] J. Gallego, M. Pardàs, and G. Haro, "Bayesian foreground segmentation and tracking using pixel-wise background model and region based foreground model," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, Nov 2009, pp. 3205–3208.

[36] T. Bouwmans, F. El Baf, and B. Vachon, "Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey," *Recent Patents on Computer Science*, vol. 1, no. 3, pp. 219–237, Nov. 2008. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00338206

[37] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2, 1999, p. 252 Vol. 2.

[38] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, Aug 2000.

[39] M. Harville, "A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models," in *Computer Vision — ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part III*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 543–560.

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

[41] Z. Zivkovic and F. van der Heijden, "Recursive unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 651–656, May 2004.

[42] H. Yang, Y. Tan, J. Tian, and J. Liu, "Accurate dynamic scene model for moving object detection," in *2007 IEEE International Conference on Image Processing*, vol. 6, Sept 2007, pp. VI – 157–VI – 160.

[43] W. Zhang, X. Fang, X. Yang, and Q. M. J. Wu, "Spatiotemporal gaussian mixture model to detect moving objects in dynamic scenes," *Journal of Electronic Imaging*, vol. 16, pp. 16 – 16 – 6, 2007.

[44] B. Zhong, H. Yao, S. Shan, X. Chen, and W. Gao, "Hierarchical background subtraction using local pixel clustering," in *2008 19th International Conference on Pattern Recognition*, Dec 2008, pp. 1–4.

[45] T. Fabian, *Mixture of Gaussians Exploiting Histograms of Oriented Gradients for Background Subtraction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 716–725.

[46] J. K. Suhr, H. G. Jung, G. Li, and J. Kim, "Mixture of gaussians-based background subtraction for bayer-pattern image sequences," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 3, pp. 365–370, March 2011.

[47] T. Bouwmans, F. El Baf, and B. Vachon, "Statistical Background Modeling for Foreground Detection: A Survey," in *Handbook of Pattern Recognition and Computer Vision*. World Scientific Publishing, Jan. 2010, pp. 181–199.

[48] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ser. ECCV '00. London, UK, UK: Springer-Verlag, 2000, pp. 751–767.

[49] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1778–1792, Nov 2005.

[50] A. Tavakkoli, M. Nicolescu, and G. Bebis, "Automatic statistical object detection for visual surveillance," in *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*, 2006, pp. 144–148.

[51] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Background modeling and subtraction by codebook construction," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 5, Oct 2004, pp. 3061–3064 Vol. 5.

[52] C. David, V. Gui, and F. Alexa, "Foreground/background segmentation with learned dictionary," in *Proceedings of the 3rd International Conference on Applied Mathematics, Simulation, Modelling, Circuits, Systems and Signals*, ser. ASMCSS'09. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 197–201.

[53] D. Mukherjee and Q. JonathanWu, "Real-timevideosegmentation using student'stmixture model," *Procedia Computer Science*, vol. 10, pp. 153 – 160, 2012, aNT 2012 and MobiWIS 2012.

[54] T. S. F. Haines and T. Xiang, "Background subtraction with dirichlet processes," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, ser. ECCV'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 99–113.

[55] T. S. F. Haines and T. Xiang, "Background subtraction with dirichletprocess mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 670–683, April 2014.

[56] J. Ding, M. Li, K. Huang, and T. Tan, *Modeling Complex Scenes for Accurate Moving Objects Segmentation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 82–94.

[57] Z. Liu, W. Chen, K. Huang, and T. Tan, "A Probabilistic Framework Based on KDE-GMM Hybrid Model for Moving Object Segmentation in Dynamic Scenes," in *The Eighth International Workshop on Visual Surveillance - VS2008*. Marseille, France: Graeme Jones and Tieniu Tan and Steve Maybank and Dimitrios Makris, Oct. 2008. [Online]. Available: https://hal.inria.fr/inria-00325761

[58] Z. Liu, K. Huang, and T. Tan, "Foreground object detection using top-down information based on em framework," *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 4204–4217, Sept 2012.

[59] O. Barnich and M. V. Droogenbroeck, "Vibe: A powerful random technique to estimate the background in video sequences," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 945–948.

[60] 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.

[61] M. V. Droogenbroeck and O. Paquot, "Background subtraction: Experiments and improvements for vibe," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 32–37.

[62] J. Dou and J. Li, "Moving object detection based on improved vibe and graph cut optimization," *Optik - International Journal for Light and Electron Optics*, vol. 124, no. 23, pp. 6081 – 6088, 2013.

[63] K. Cheng, K. Hui, Y. Zhan, and M. Qi, "A novel improved vibe algorithm to accelerate the ghost suppression," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Aug 2016, pp. 1692–1698.

[64] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 38–43.

[65] H.-H. Lin, T.-L. Liu, and J.-H. Chuang, "A probabilistic svm approach for background scene initialization," in *Proceedings. International Conference on Image Processing*, vol. 3, June 2002, pp. 893–896 vol.3.

[66] H. H. Lin, T. L. Liu, and J. H. Chuang, "Learning a scene background model via classification," *IEEE Transactions on Signal Processing*, vol. 57, no. 5, pp. 1641–1654, May 2009.

[67] P. Jiménez, S. Maldonado-Bascón, R. Gil-Pita, and H. Gómez-Moreno, *Background Pixel Classification for Motion Detection in Video Image Sequences*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 718–725.

[68] J. Zhao, *Segmenting Moving Objects with a Recurrent Stochastic Neural Network*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 666–672.

[69] J. Zhao, *Dynamic Background Discrimination with a Recurrent Network*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 432–437.

[70] L. Maddalena and A. Petrosino, *A Self-organizing Approach to Detection of Moving Patterns for Real-Time Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 181–190.

[71] L. Maddalena and A. Petrosino, *Neural Model-Based Segmentation of Image Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 57–64.

[72] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168–1177, July 2008.

[73] L. Maddalena and A. Petrosino, *3D Neural Model-Based Stopped Object Detection*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 585–593.

[74] K. Sehairi, F. Chouireb, and J. Meunier, "Comparative study of motion detection methods for video surveillance systems," *Journal of Electronic Imaging*, vol. 26, pp. 26 – 26 – 29, 2017.

[75] P. Xu, M. Ye, X. Li, Q. Liu, Y. Yang, and J. Ding, "Dynamic background learning through deep auto-encoder networks," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 107–116.

[76] P. Xu, M. Ye, Q. Liu, X. Li, L. Pei, and J. Ding, "Motion detection via a couple of auto-encoder networks," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, July 2014, pp. 1–6.

[77] M. Braham and M. V. Droogenbroeck, "Deep background subtraction with scene-specific convolutional neural networks," in *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*, May 2016, pp. 1–4.

[78] C. M. Bautista, C. A. Dy, M. I. Mañalac, R. A. Orbe, and M. Cordel, "Convolutional neural network for vehicle detection in low resolution traffic videos," in *2016 IEEE Region 10 Symposium (TENSYMP)*, May 2016, pp. 277–281.

[79] H. Al-Khateeb and M. Petrou, "Automatic change detection in an indoor environment," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, June 2010, pp. 53–58.

[80] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in real-time," in *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 4, Dec 1994, pp. 3776–3781 vol.4.

[81] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979.

[82] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[83] T. Lim, S. Hong, B. Han, and J. H. Han, "Joint segmentation and pose tracking of human in natural videos," in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 833–840.

[84] H.-G. Kang and S.-H. Lee, "Human body posture recognition with discrete cosine transform," in *2016 International Conference on Big Data and Smart Computing (BigComp)*, Jan 2016, pp. 423–426.

[85] S. Li, H. Lu, and X. Shao, "Human body segmentation via data-driven graph cut," *IEEE Transactions on Cybernetics*, vol. 44, no. 11, pp. 2099–2108, Nov 2014.

[86] A. M. Sharma, K. Venkatesh, and A. Mukerjee, "Human pose estimation in surveillance videos using temporal continuity on static pose," in *2011 International Conference on Image Information Processing*, Nov 2011, pp. 1–6.

[87] H. Ji and F. Su, "Robust 3d human pose estimation via dual dictionaries learning," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov 2012, pp. 3370–3373.

[88] W. Gong, P. Fihl, J. Gonzalez, T. B. Moeslund, W. Zhang, Z. Li, and Y. Ren, "Robustness of input features from noisy silhouettes in human pose estimation," in *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*, Oct 2014, pp. 126–131.

[89] T. Huynh-The, B.-V. Le, S. Lee, and Y. Yoon, "Interactive activity recognition using pose-based spatio–temporal relation features and four-level pachinko allocation model," *Information Sciences*, vol. 369, pp. 317 – 333, 2016.

[90] Y. Dong, D. Conrad, and G. N. DeSouza, "Using background subtraction and human pose recognition to eliminate game controllers," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3887–3892.

[91] P. Afsar, P. Cortez, and H. Santos, "Automatic human action recognition from video using hidden markov model," in *2015 IEEE 18th International Conference on Computational Science and Engineering*, Oct 2015, pp. 105–109.

[92] W. Lin, M.-T. Sun, R. Poovandran, and Z. Zhang, "Human activity recognition for video surveillance," in *2008 IEEE International Symposium on Circuits and Systems*, May 2008, pp. 2737–2740.

[93] X. Yuan and X. Yang, "A robust human action recognition system using single camera," in *2009 International Conference on Computational Intelligence and Software Engineering*, Dec 2009, pp. 1–4.

[94] S. Kumari and S. K. Mitra, "Human action recognition using dft," in *2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, Dec 2011, pp. 239–242.

[95] J. Wang, R. Chen, X. Sun, M. She, and L. Kong, “Generative models for automatic recognition of human daily activities from a single triaxial accelerometer,” in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1–6.

Appendix A

List of Publications

International Journal Papers

[1] **Thien Huynh-The**, Cam-Hao Hua, Tu Anh Nguyen, Taeho Hur, Jaehun Bang, Dohyeong Kim, Muhammad Bilal Amin, Byeong Ho Kang, Hyonwoo Seung, Sungyoung Lee, "Selective Bit Embedding Scheme For Robust Blind Color Image Watermarking," Information Sciences (SCI, IF: 4.832), 2017 (Accepted).

[2] **Thien Huynh-The**, Cam-Hao Hua, Tu Anh Nguyen, Taeho Hur, Jaehun Bang, Dohyeong Kim, Muhammad Bilal Amin, Byeong Ho Kang, Hyonwoo Seung, Sungyoung Lee, "Hierarchical Topic Modeling With Pose-Transition Feature For Action Recognition Using 3D Skeleton Data," Information Sciences (SCI, IF: 4.832), 2017 (Major Revision).

[3] Dinh-Mao Bui, **Thien Huynh-The** and Sungyoung Lee, "Early fault detection in IaaS cloud computing based on fuzzy logic and prediction technique," The Journal of Supercomputing (SCI, IF: 1.088), April 2017.

[4] **Thien Huynh-The**, Oresti Banos, Sungyoung Lee, Byeong Ho Kang, Eun-So Kim, Thuong Le-Tien, "NIC: A Robust Background Extraction Algorithm for Foreground Detection in Dynamic Scenes," IEEE Transactions on Circuits and Systems for Video technology (SCI, IF: 3.599), vol. 27, no. 7, pp. 1478-1490, July 2017.

[5] Oresti Banos, Claudia Villalonga, Jaehun Bang, Taeho Hur, Dongwook Kang, Sangbeom Park, **Thien Huynh-The**, Ba-Vui Le, Muhammad Bilal Amin, Muhammad Asif Razzaq, Wajahat Ali Khan, Choong Seon Hong, Sungyoung Lee, "Human Behavior Analysis by means of Multimodal Context Mining," Sensors (SCIE, IF: 2.033), vol. 16, issue 8, 2016.

[6] **Thien Huynh-The**, Ba-Vui Le, Sungyoung Lee and Yongik Yoon, "Interactive Activity Recognition Using Pose-based Spatio-Temporal Relation Features and Four-Level Pachinko Allocation Model," Information Science (SCI, IF: 3.364), vol. 369, pp.317-333, 2016.

[7] **Thien Huynh-The**, Oresti Banos, Sungyoung Lee, Yongik Yoon, Thuong Le-Tien, "Improving Digital Image Watermarking by Means of Optimal Channel Selection," Expert Systems With Applications (SCIE, IF: 2.981), vol. 62, pp. 177-189, 2016.

[8] **Thien Huynh-The**, Oresti Banos, Ba-Vui Le, Dinh-Mao Bui, Yongik Yoon, Sungyoung Lee, "Traffic Behavior Recognition Using Pachinko Allocation Model", Sensors (SCIE, IF: 2.245), vol.15, issue 7, pp.16040-16059,2015.

[9] **Thien Huynh-The**, Ba-Vui Le, Sungyoung Lee, Thuong Le-Tien and Yongik Yoon, "Using Weighted Dynamic Range for Histogram Equalization to Improve The Image Contrast," EURASIP Image and Video Processing (SCIE, IF: 0.662), vol.44, no.10, pp.1-17, 2014.

International Conference Papers

[1] **Thien Huynh-The**, Sungyoung Lee, and Cam-Hao Hua, "ADM-HIPaR: An Efficient Background Subtraction Approach," 2017 IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS 2017), Aug 29- Sep 1, 2017, Lecce, Italy.

[2] **Thien Huynh-The** and Sungyoung Lee, "Color Image Watermarking Using Selective MSB-LSB Embedding And 2D Otsu Thresholding," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2017), Oct. 5-8, 2017, Banff, Canada.

[3] **Thien Huynh-The**, Cam-Hao Hua, Sungyoung Lee, "Improving NIC Algorithm Using Different Binary Structure Elements For Multi-modal Foreground Detection," In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication (IMCOM ′17), Jan 5-7, Beppu, Japan, 2017.

[4] **Thien Huynh-The**, Ba-Vui Le, Sungyoung Lee, "Describing Body-Pose Feature - Poselet - Activity Relationship Using Pachinko Allocation Model," 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016), Oct. 9-12, 2016, Budapest, Hungary.

[5] Dinh-Mao Bui, **Thien Huynh-The**, and Sungyoung Lee, "Fuzzy Fault Detection in IaaS Cloud Computing," In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication (IMCOM ′16), Jan 4-6, Danang, Vietnam, 2016.

[6] Thuong Le-Tien, **Thien Huynh-The**, Sungyoung Lee, and Yongik Yoon, "An Interactive Activity Recognition Approach Using Articulated-Body Estimation and Pose-Based Features," InProceedings of the 10th International Conference on Ubiquitous Information Management and Communication (IMCOM ′16), Jan 4-6, Danang, Vietnam.

[7] **Thien Huynh-The**, Dinh Mao Bui, Sungyoung Lee and Yongik Yoon, "Improved PAM-based traffic behavior recognition using trajectory-wise features," 2016 International Conference on Big Data and Smart Computing (BigComp), Hong Kong, 2016.

[8] Dinh Mao Bui, **Thien Huynh-The**, Sungyoung Lee, Y. Yoon and S. Jun, "Energy savings in processor based on prediction technique," 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 2016.

[9] Dinh Mao Bui, **Thien Huynh-The**, Sungyoung Lee, B. Li and J. Wang, "Replication Management Framework for HDFS Based on Prediction Technique," 2015 Third International Conference on Advanced Cloud and Big Data, Yangzhou, China, 2015.

[10] Dinh-Mao Bui, **Thien Huynh-The**, YongIk Yoon, SungIk Jun and Sungyoung Lee, "EAP: Energy-Awareness Predictor in Multicore CPU," The 10th KIPS International Conference on Ubiquitous Information Technologies and Applications (CUTE 2015) Cebu, Philippines, Dec 15-17, 2015.

[11] **Thien Huynh-The**, Dinh-Mao Bui, Sungyoung Lee, Yongik Yoon, "Interactive Activity Recognition Using Articulated-Pose Features on Spatio-Temporal Relation," The 10th KIPS International Conference on Ubiquitous Information Technologies and Applications (CUTE 2015), Cebu, Philippines, Dec 15-17, 2015.

[12] Oresti Banos, Jaehun Bang, Taeho Hur, Muhammad Hameed Siddiqi, **Thien Huynh-The**, Le-Ba Vui, Wajahat Ali Khan, Taqdir Ali, Claudia Villalonga and Sungyoung Lee, "Mining

human behavior for health promotion," 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, 2015.

[13] Dinh Mao Bui, **Thien Huynh-The**, Sungyoung Lee and Yongik Yoon, "Complexity reduction for Gaussian process regression in spatio-temporal prediction," Advanced Technologies for Communications (ATC), 2015 International Conference on, Ho Chi Minh City, 2015.

[14] **Thien Huynh-The**, Oresti Banos, Ba-Vui Le, Dinh-Mao Bui, Sungyoung Lee, Yongik Yoon and Thuong Le-Tien, "PAM-based flexible generative topic model for 3D interactive activity recognition," Advanced Technologies for Communications (ATC), 2015 International Conference on, Ho Chi Minh City, 2015.

[15] **Thien Huynh-The**, Oresti Banos, Ba-Vui Le, Dinh-Mao Bui, Sungyoung Lee, Yongik Yoon and Thuong Le-Tien, "Background subtraction with neighbor-based intensity correction algorithm," Advanced Technologies for Communications (ATC), 2015 International Conference on, Ho Chi Minh City, 2015.

[16] **Thien Huynh-The**, Oresti Banos, Sungyoung Lee, Yongik Yoon, and Thuong Le-Tien, "A novel watermarking scheme for image authentication in social networks," In Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication(IMCOM ′15), Bali, Indonesia, Jan 8-10, 2015.

[17] **Thien Huynh-The**, Ba-Vui Le, Muhammad Fahim, Sungyoung Lee, Yongik Yoon, and Byeong Ho Kang, "PAM-based Behavior Modelling," Ambient Assisted Living and Daily Activities: 6th International Work-Conference, IWAAL 2014, Belfast, UK, Dec 2-5, 2014.

[18] **Thien Huynh-The**, Sungyoung Lee, Pham Chi Hieu and Thuong Le-Tien, "A DWT-based image watermarking approach using quantization on filtered blocks," 2014 International Conference on Advanced Technologies for Communications (ATC 2014), Hanoi, 2014.

[19] **Thien Huynh-The**, Hai Nguyen Thanh and Hung Tran Cong, "Traffic Sign Recognition using Multi-class morphological detection," 2014 International Conference on Advanced Technologies for Communications (ATC 2014), Hanoi, 2014.

International Patents

1. Sungyoung Lee, Thien Huynh-The , ”Watermark Insertion AND Detection Apparatus and Method Thereof”, Application No. PCT/KR2016/005770, May 31, 2016.