Hysteresis-Based Selective Gaussian-Mixture Model for Real-Time Background Update and Object Detection

Firas Achkar

A Thesis in The Department of Electrical and Computer Engineering

Presented in Partial Fulfillment of The Requirements
for The Degree of Master of Applied Science at

Concordia University

Montréal, Quebéc, Canada
November 8, 2006

Your file Votre référence
ISBN: 978-0-494-28908-2
Our file Notre référence
ISBN: 978-0-494-28908-2

Abstract

Hysteresis-Based Selective Gaussian-Mixture Model for Real-Time Background Update and Object Detection

Firas Achkar

Background subtraction refers to background update and object detection, and it is a commonly used object segmentation technique. In this technique a background model frame is built and updated over time such that it only corresponds to static pixels of the monitored scene. Moving objects are then detected by subtracting each new frame from this background model frame.

In this thesis, we propose two real-time effective techniques for video object segmentation: the first is a background subtraction technique that includes background update and object detection stages to extract object binary blobs; the second is an improved contour tracing and a new filling algorithms to extract object features such as area, compactness and irregularity. The proposed background subtraction technique effectively models the static background and detects true moving objects while retaining computational efficiency for the real-time criteria.

In the background update stage of the proposed background subtraction technique, the reference background pixels are modeled as multiple color Gaussian distributions (MOGs) with a new selective matching scheme based on the combined approaches of component ordering and winner-takes-all. This matching scheme not only selects the most probable component for the first matching with new pixel data, greatly improving performance, but also simplifies pixel classification and component replacement in case of no match. Further performance improvement to background update stage is achieved by using a new simple yet functional component variance adaptation formula. A periodical weight normalization scheme is used to prevent merging temporary stopped real foreground object into the background model, and the creation of false ghosts in the foreground mask when these objects start to move again. The proposed background update technique implicitly handles both gradual illumination change and temporal clutter problems.

The object detection stage uses two schemes that improve object blob quality: a new hysteresis-based component matching to reduce the amount of cracks and added shadows; and temporal motion history to preserve the integrity of moving object boundaries. In this stage, the problem of shadows and ghosts is partially addressed by the proposed hysteresis-based matching scheme, while the problems of persistent sudden illumination changes and camera perturbations are addressed at frame level depending on the percentage of pixels classified as foreground.

After background subtraction the detected moving object pixels (initial foreground binary mask) are highly abstract and must be grouped together to form the actual objects. We propose an improved contour tracing and new filling algorithms for grouping object pixels. The proposed improved tracing algorithm can detect and reject dead or inner branches, false non-closed contours, noise related small contours, and then efficiently categorize each contour into inner or outer contours. The new filling algorithm is efficient and never leaks, it uses the extracted contour points and their chain-code information as seed points for horizontal line growing. Experimental results show that the proposed tracing and filling techniques improve computational performance with no tracing or filling errors compared to other reference techniques.

Acknowledgments

I wish to acknowledge all of the people who have helped me with this thesis. I would like to express my gratitude to Assist. Prof. Aishy Amer for her supervision, continues guidance and support. I would like also to thank both Prof. Zuheir Wakkaf and Prof. Mariam Saai for their profound influence on my scientific engineering approach for addressing and solving problems.

Contents

List of Figures ix

List of Tables xi

1 Introduction 1

1.1 Definitions 2

1.2 Objective and Problem Statement 3

1.3 Overview of Proposed Techniques 4

1.4 Contributions 5

1.4.1 Background Update and Object Detection 5

1.4.2 Contour Tracing and Filling 6

1.5 Thesis Organization 7

2 Review of Background Subtraction Techniques 8

2.1 Non-recursive Techniques 10

2.1.1 Simple Frame Differencing 10

2.1.2 Median Filter 10

2.1.3 Linear Predictive Filter 12

2.1.4 Non-Parametric Model 12

2.1.5 Other Less Promising Non-recursive Techniques 14

2.2 Recursive Techniques 15

2.2.1 Approximated Temporal Median Filter 15

2.2.2 Kalman Filter and Single Gaussian 15

2.2.3 Mixture of Gaussians (MOG) 17
2.3 Summary 19

3 Proposed Background Subtraction (Background Update and Object Detection) 20

3.1 Notation 21
3.2 Model Formation 23
3.3 Background Update 25
3.3.1 Selective Component Matching 25
3.3.2 Component Parameter Update 26
3.4 Object Detection 26
3.4.1 Motion History 27
3.4.2 Pixel Classification 27
3.4.3 Sudden Illumination Detection 27
3.5 System Implementation and Parameters 28
3.6 Analysis and Comparison to MOG Methods 29
3.6.1 Contributions Analysis 29
3.6.2 Addressed Problems 31
3.6.3 Computational Performance Analysis 32
3.7 Analysis and Comparison to Non-MOG Techniques 35
3.8 Summary 36

4 Experimental Results of Proposed Background Subtraction 38

4.1 Objective Evaluation 39
4.2 Subjective Evaluation 40
4.2.1 "Fog" Sequence 40
4.2.2 "Winter" Sequence 44
4.2.3 "Snowing" Sequence 44
4.2.4 "Rene-Levesque Guy" Sequence 44
4.2.5 "St-Catherine" Sequence 49
4.3 Computational Performance 49

4.4 Practical Considerations 52

5 Contour Tracing and Filling **55**

5.1 Introduction 55

5.2 A Survey of Related Work 57

5.3 Improved Contour Tracing Algorithm 59

5.3.1 Notation 60

5.3.2 Module Tracing 61

5.3.3 Module Tracer 62

5.3.4 Module Delete Dead Branch 63

5.3.5 Module Connected Closed Contour 63

5.3.6 Module Check Internal 65

5.4 Proposed New Filling Algorithm 66

5.5 Analysis of Proposed Contour Tracing and Filling 67

5.5.1 Functional analysis 68

5.5.2 Complexity and efficiency 72

5.5.3 System Implementation and Parameters 73

5.6 Experimental Results 74

5.6.1 Objective evaluation 75

5.6.2 Subjective evaluation 77

5.7 Summary 82

6 Conclusion and Future Work **84**

6.1 Conclusion 84

6.2 Future Work 85

Bibliography **87**

A List of Symbols and Abbreviations **95**

A.1 List of Symbols 95

A.2 Abbreviations 98

B Algorithms Bench Application **99**

B.1 Description 99

B.2 Installation and Usage Guide 100

B.3 On-Line System Operation 101

List of Figures

1.1 Moving object segmentation. 2

2.1 Background subtraction flow diagram (with tracing and filling). . . . 9

3.1 Block diagram of the proposed background update and object detection technique. 21

3.2 Component tracking of pixel intensity in time. 24

3.3 $G_{k,t}$ components overlap for a given pixel p_c. 26

3.4 The dual problem of merged foreground objects and the subsequent false ghosts in most MOG techniques. 33

3.5 Shadow and ghost reduction using the proposed hysteresis thresholding scheme. All parameters are fixed except for λ_1, λ_2. 34

4.1 Euclidean distance objective comparison for "Hall" sequence. 40

4.2 Euclidean distance objective comparison for "Intelligentroom" sequence. 41

4.3 Algorithm outputs for sample frame I_{110} of "Fog" sequence. 42

4.4 Algorithm outputs for sample frame I_{210} of "Fog" sequence. 43

4.5 Algorithm outputs for sample frame I_{119} of "Winter" sequence. 45

4.6 Algorithm outputs for sample frame I_{249} of "Winter" sequence. 46

4.7 Algorithm outputs for sample frame I_{107} of "Snowing" sequence. . . . 47

4.8 Algorithm outputs for sample frame I_{214} of "Snowing" sequence. . . . 48

4.9 Algorithm outputs for the "Rene-Levesque Guy" crossing sequence. . 50

4.10 Algorithm outputs for the "St-Catherine" sequence. 51

4.11 Average computational performance chart for the first 100 over all the test sequences. 52
4.12 Foreground objects of 10% frame size. 53
4.13 Foreground objects larger than 20% frame size. 53
4.14 Very fast moving truck. 54

5.1 The neighborhood of a pixel p_c. 56
5.2 Proposed tracing algorithm block diagram. 60
5.3 Scanning Direction. 61
5.4 Module-Tracer block diagram. 63
5.5 Module Trace sub-cases illustrated. 64
5.6 Special case contours illustrated. 64
5.7 Internal checking mechanism in Module Check Internal. 66
5.8 Filling cases illustrated. 67
5.9 Un-efficient next contour points search mechanism by Pavlidis. 68
5.10 Crossed contour points of a real video object. 69
5.11 Complex contours indicated on the "Synthetic Image" and sample frames of "Hall" and "Meetingroom" sequences. 74
5.12 Objective comparison of tracing and filling outputs for "Synthetic Image". 76
5.13 Subjective comparison for "Synthetic Image". 76
5.14 Objective comparison for "Hall" sequence. 77
5.15 Subjective comparison for "HALL" sequence. 78
5.16 Objective comparison for "Meetingroom" sequence. 78
5.17 Subjective comparison for "Meetingroom" sequence. 79
5.18 "Hall" sequence: non-representative tracing by reference methods of dead or inner branches and connected small contours. 80
5.19 Algorithm outputs for frame sample $E(291)$ of "Hall" sequence. . . . 81
5.20 Algorithm outputs for frame sample $E(6)$ of "Survey" sequence. . . . 82

B.1 Screen shot of the test bench application. 100

List of Tables

5.1 Average tracing and filling times in seconds per frame. 73

Chapter 1

Introduction

The advances in computing power, availability of large-scale storage devices and high bandwidth network infra-structures lead to the wide spread of video-based applications not only in research but in various fields of our real world such as automated industrial quality verification, robotic arm vision, Internet and TV broadcast streaming, and video surveillance.

Most video-based applications consist of a low-level pixel-based object segmentation stage, e.g., detecting object binary blobs or extracting object features such as width, height, area, compactness and irregularity ratio.

Moving object detection is the most important stage in any content-based video-processing based subsystem. The more accurate and reliable the moving object detection is, the more effective and less complex the subsequent stages will be.

Precise moving object detection is a challenging scientific problem with many promising real world applications, thus drawing the attention of many researchers, institutes and companies. To detect objects in a changing environment a background update method is required. In this thesis we study the background subtraction (i.e., background update and object detection) technique. Our motivation in studying adaptive background-subtraction based object detection is to address the most important related problems [1] such as gradual ambient-illumination change, temporal clutter, bootstrapping, sleeping person, shadows and ghosts, while retaining the real-time functioning criteria necessary for any practical video system.

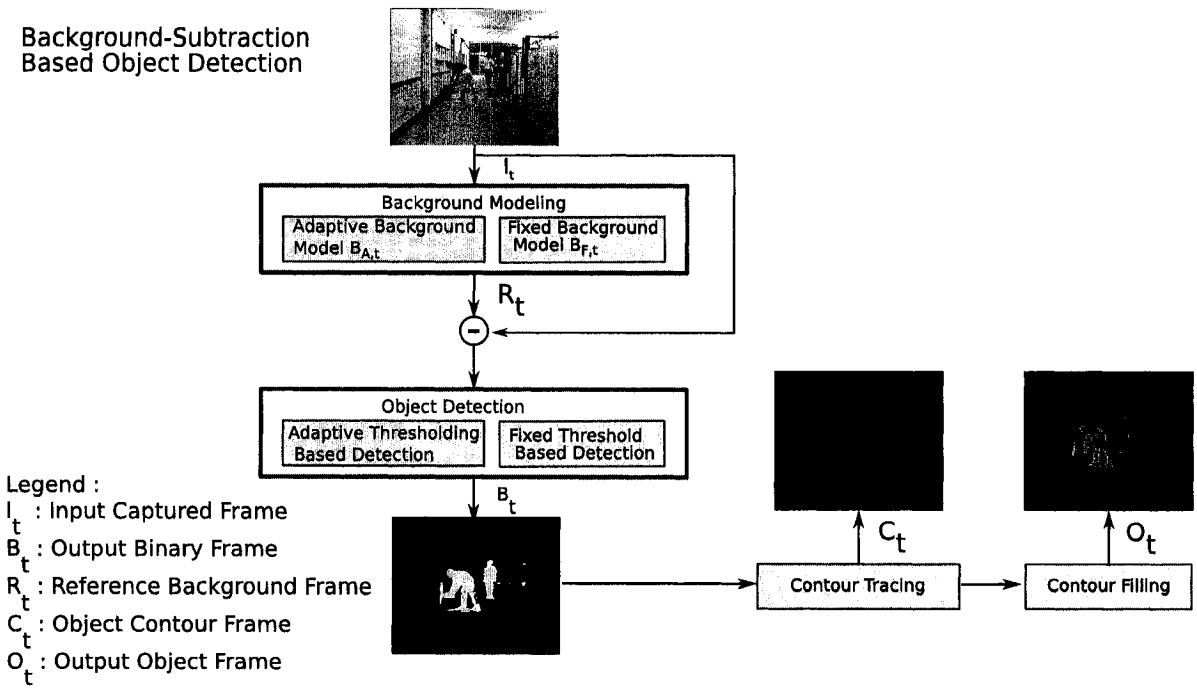


Figure 1.1: Moving object segmentation.

1.1 Definitions

Moving object segmentation (Fig. 1.1) refers to the process of grouping pixels into video objects. For an input video scene, background modeling corresponds to building a frame representation of only the stationary object pixels in that scene. Background subtraction (BS) includes two steps background update and object detection. Background update corresponds to temporal evolution and ambient change compensation of background models. Moving object detection by BS is per-pixel difference between each new frame and those corresponding pixels of the background model. Gaussian component is Gaussian distribution based statistical modeling of pixel intensity values. MOG is a per pixel Mixture (or multiple) of Gaussian components.

Contour tracing is an algorithm that groups *neighborhood* connected pixels in a binary edge image. Contour filling fills the inner pixels of a contour with specific gray-scale values.

1.2 Objective and Problem Statement

Various background update and object detection techniques have been presented in the literature for detecting moving objects. We investigate those techniques that are both effective and real-time or nearly real-time. We specifically focus on those techniques in which one or more background models are estimated and evolved frame by frame, and where moving objects are then detected by subtracting the current frame with one or more of these background models. Backgrounds usually consists of objects that remain passive in the scene. These objects can be either stationary static pixels, such as walls, doors, room furniture, traffic signs, or non-stationary dynamic pixels, such as wavering bushes, moving escalators, or most commonly environmental conditions of rain, snow and ambient brightness. In BS moving objects are detected by subtracting the current frame with one or more of those background models.

BS has broadly two problems usually related to its dynamic pixels. The first problem is that the background models (at least one of them) should reflect the current real background; this means addressing the following problems [1]: time of day, camouflage, sleeping person, bootstrapping and waving trees. The second problem is that the background model should be adaptive and immediately handle sudden scene changes such as moved object, light switch and shadows (and ghosts). Our main objective in this thesis is to address the previous problems of BS achieving precise non-background dynamic moving object detection.

The detected moving object pixels (initial foreground binary mask) are highly abstract and must be grouped together to form the actual objects. Contour tracing and filling based techniques are widely used techniques to extract objects from binary blobs. They adhere to the real-time and limited storage criteria necessary for practical on-line video processing applications.

Classical still-image based contour tracing and filling algorithms blindly trace and fill contours without consideration for their representativeness to true video objects. This leads to many subtly non-corresponding contours that cause unpredictable behaviors and failures in subsequent higher object-processing stages. Our secondary

objective in this thesis is to achieve representative video object tracing by checking the correctness of video contour shapes at pixel level and amending any cases where unwanted distortions (e.g. inner branches) may occur at each new candidate contour pixel. Note that after tracing and filling, various features such as width, height, area, compactness, and irregularity ratio of each traced and labeled video object can be easily extracted. These features then can be used in the following higher processing stages of video object tracking, event analysis and content-based video coding.

1.3 Overview of Proposed Techniques

In this thesis, we propose an adaptive BS-based technique for moving object detection and an improved video object labeling technique. The proposed techniques operate on both gray-scale and color video streams obtained from any indoor or outdoor fixed visual sensor.

In the proposed adaptive BS-based object detection technique, each pixel of the adaptive background-model frame is modeled as MOG. In the background update stage, MOG components are always descendingly stored according to their accumulated weight values. New pixel data are compared with MOG components using a hysteresis-based selective matching scheme. The mean, variance and weight values of the matched component are recursively updated with the new pixel data. Component update insures that the different components track changing background and foreground intensity variations, only adapting to the most persistent in a predefined time slice. Also in this stage, component weights are periodically normalized to emphasizing the effect of more recent pixel data over older ones.

In the object detection stage, motion history and weight value of the matched component are used to classify new pixel data to either background or foreground. In the frame level of this stage, sudden illumination changes are detected and handled based on the percentage of pixels classified as foreground.

The proposed improved contour tracing algorithm takes as an input a binary edge image. The tracing starts by locating, in a contra-clockwise manner, all 8-neighbor

connected closed contours in a video frame. And while tracing, if a visited non-start pixel (complex contour) is detected with a given set of conditions corresponding to dead, inner branches or non closed contours, our algorithm removes these traced pixels, keeping only true video contours.

The proposed new contour filling algorithm combines the simplicity of a label-based and the robustness of seed growing techniques. Labels are obtained during the tracing phase, corresponding to the chain-code or directions of each contour point. Then these points in a given scan-line of the contour are used as either a terminating or line growing seeds depending on their label information, insuring that the algorithm can handle all subtle cases of complex concavities, crossing edge points and parallel edges. Also in this algorithm, specific features of each contour such as width, height, are, compactness and irregularity ratio are computed while tracing the edge image, and later stored with the extracted contour points as independent objects for subsequent processing stages.

1.4 Contributions

1.4.1 Background Update and Object Detection

To the knowledge of the author at the time the proposed schemes of this thesis were developed, the key original contributions of the proposed background update and object detection technique[1], compared to the techniques in [3–9] are:

- New selective matching scheme based on the combined approaches of component ordering and winner-takes-all. This matching scheme not only selects the

[1] A conference paper based on the proposed background update and object detection technique was accepted for publication: "Hysteresis-Based Selective Gaussian-Mixture Model for Real-Time Background Maintenance", IS&T/SPIE Symposium on Electronic Imaging, Conference on Visual Communications and Image Processing, January 2007, San Jose, CA, US. Also a journal paper based on the proposed background update and object detection technique was submitted for publication: "Real-Time Background Update Using Hysteresis-Based Selective Gaussian-Mixture Model", Springer Trans. on Signal, Image and Video Processing special issue, March 2007. In addition, a journal paper "Accurate Tracing and Filling of Complex Contours for Object-Based Video Processing" based on [2] and the improvements proposed to [2] in this thesis has been submitted to IEEE Trans. on Image Processing, 2007.

most probable component for the first matching with new pixel data, greatly improving performance, but also simplifies pixel classification and component replacement in case of no match.

- New simple component parameter adaptation formulas that results in further performance improvement.

- New embedded hysteresis-based component matching and temporal motion history schemes that improve object detection accuracy. Component hysteresis matching improves detected foreground object blobs by reducing the amount of cracks and added shadows, while motion history preserves the integrity of moving objects boundaries, both with minimum computational overhead.

In addition, various reference methods and techniques [1, 3–22] of adaptive-model based BS techniques were studied, three state-of-the-art adaptive techniques [8,23,24] were implemented, and their performance analyzed and compared with the proposed background update technique.

1.4.2 Contour Tracing and Filling

The proposed tracing algorithm is an improvement of the original work of [2], while the proposed filling algorithm is novel. The key original contributions of the proposed contour tracing and filling technique compared to [25–29] are:

- Improved computationally efficient contour tracing algorithm for handling cross-point connected contours, making this algorithm fast, robust and less error prone.

- New fast and effective contour filling algorithm.

In addition, various reference tracing and filling techniques [2, 25–31] were studied, two reference binary tracing and filling techniques [25, 29] were implemented, and their performance analyzed and compared with the proposed contour tracing and filling algorithms.

1.5 Thesis Organization

The thesis is organized as following. In Chapter 2, we present a review of background subtraction techniques. In Chapter 3, we present the proposed background update and object detection techniques. In Chapter 4 we present the experimental results and comparison of the proposed background subtraction method. In Chapter 5, we present the proposed tracing and filling techniques. Finally Chapter 6 concludes the thesis with the suggestions for future work.

Chapter 2

Review of Background Subtraction Techniques

In background subtraction, each arriving new video frame is compared against a reference background model. Pixels in the current frame that differ significantly from the corresponding background model pixels are considered to be foreground moving objects.

Although many background subtraction algorithms have been presented in literature, roughly they all share the same four stage structure of: preprocessing, background update, subtraction (object detection), and post-processing (Fig. 2.1). The preprocessing stage may include temporal and/or spatial smoothing to reduce camera and transient environmental noise, global motion compensation to cancel the effect of sudden camera perturbation, and histogram or color normalization to reduce ambient illumination change. In the background update stage, the background model is updated to reflect the latest acquired data. A pixel-wise thresholding is preformed in the subtraction stage producing the initial foreground mask. In the final stage each moving object is extracted from the foreground mask and stored in a suitable format for subsequent higher processing stages.

Despite that background subtraction and update is very active field of research in video processing, computer vision and robotics there is no unified common methodology for classifying those techniques. Many works in literature try, within their

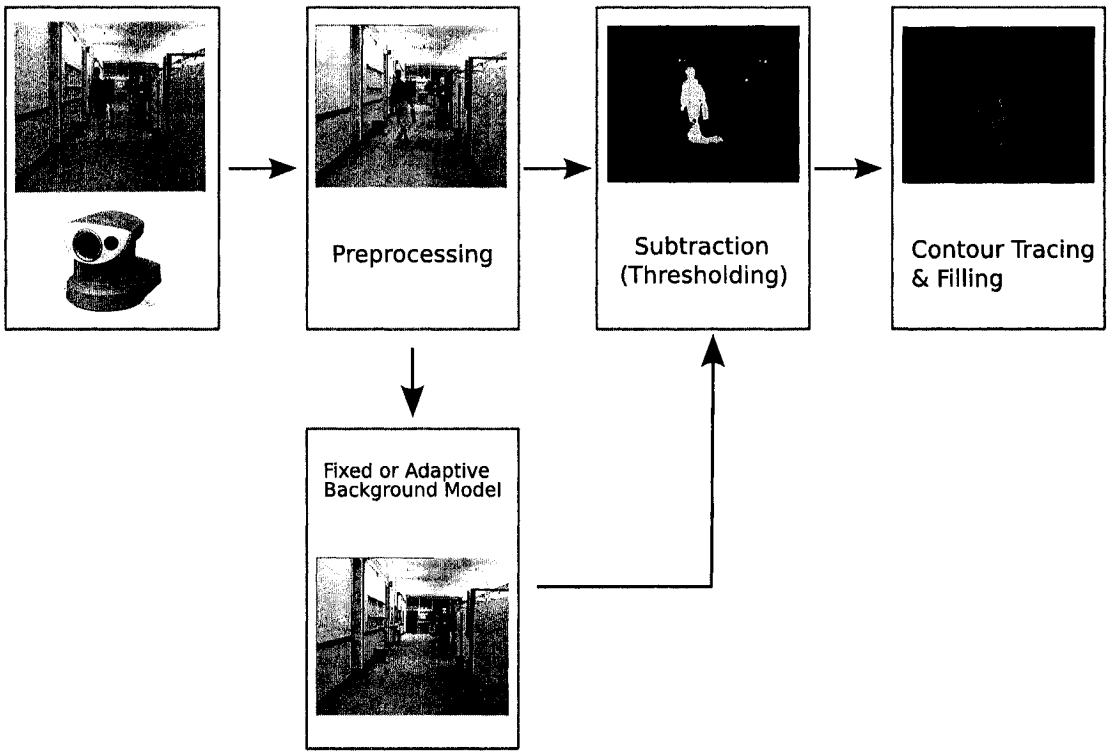


Figure 2.1: Background subtraction flow diagram (with tracing and filling).

introduction or related work parts, to categorize moving object detection or background subtraction/update techniques in such a way to emphasize their presented techniques. For instance Liyuan et al. [20] categorized background update techniques based on their information domain; spectral, spatial or temporal. Cucchiara et al. [23] tried to categorize previous work according to the problems these techniques address. Suchendra et al. [22] focused on the sleeping person problem and grouped previous work to those that handle this issue and those do not.

We review and categorize previous work based on background modeling scheme and the technique used to update it; also we emphasize the advantages of each technique and what problems it fails to address. We found a review reference by Piccardi [19] in which the author viewed various techniques based on their modeling/updating techniques, including relative comparisons of computational complexity, memory requirement and output accuracy of these techniques. Also, we found two other reports [11, 16] with better content and details, in which they again categorize the most effective methods based on the background modeling and the problems they

address. We present the various methods classifying them into two broad categories; non-recursive and recursive.

Note that many of the papers we reviewed do not clearly differentiate between background update and object detection but call both steps background update.

2.1 Non-recursive Techniques

A non-recursive technique stores a buffer of previous L video frames, and estimates the background model based on per pixel buffer temporal variation. Non-recursive techniques are highly adaptive but may require significant storage if a large buffer is needed to cope with the subtle slow moving objects. Following are the most common in this category:

2.1.1 Simple Frame Differencing

In this technique, one or two previous frames are used as background models and the current frame is differenced from those background models. In this technique, usually no background update is utilized (non-adaptive). Frame differencing alone is not very effective since it fails in detecting the interior pixels of a uniformly-colored moving objects. It will also totally fail and produce no significant output if the moving objects simply stop moving. This technique although as mentioned is not affective by itself, when used with other techniques, it can help to address various problems like moved objects, camouflage and a waking person . The works of [1, 20, 22, 32–34] are good examples where frame differencing is used in combination with other techniques.

2.1.2 Median Filter

Median filtering is a very common, fast yet effective background modeling technique [17,23,35,36]. In this technique, the background model (adaptive) is defined to be the temporal median at each pixel of all the L frames in the buffer, with typical values for L range from 50-200. In the worst case, each pixel has to contain back-

ground content for at least half the buffer length, otherwise the median will not be the mean background luminance or chrominance, but a shifted version of the real background pixel values. Foreground pixels (object detection) are marked by thresholding the difference between current frame pixels and those of the background model. The median-filter based techniques [17,23,35,36] can effectively handle: time of day, sleeping person and noise related problems. However, since the adaptation of background model pixels require at least $L/2$ exposure time, median-filter based techniques fail in quickly addressing: bootstrapping, light switch, waving trees, moved object (waking person), and shadows/ghosts. The work of [23] is a well detailed, latest good representative of this category. They incorporated into the median-based background model update:

- Adaptive weighting factor ω_b in the median calculation

$$S^t(I(p)) = \{I^t_{RGB}(p),\ I^{t-\Delta t}_{RGB}(p), \ldots, I^{t-t\Delta t}_{RGB}(p)\} \bigcup \omega_b\{B^t(p)\}, \qquad (2.1)$$

  where $S^t(I(p))$ is the set of buffer intensity values I_{RGB} for the pixel p at location $(i,\ j)$ at time t, $I_{RGB}(p)$ is the intensity values of a pixel p at location $(i,\ j)$ in the L length buffer, $B^t(p)$ is background model at frame t, ω_b an adequate weight that improve the stability of the background model.

- Selective updating through object-level-reasoning (higher level processing stages) for reducing object detection errors especially in the case of moved object and sleeping person problems.

- shadow and ghosts detection through the use of chromacity, optical flow and the higher logic object analysis.

In general, this BS-based modeling technique [23] without the higher level processing is computationally efficient with high storage requirement and can be quite effective.

2.1.3 Linear Predictive Filter

The background model (adaptive) is computed by applying a linear predictive filter on each corresponding pixel in the buffer. The only work we found that used this kind of technique in the literature is Wallflower of Toyama et al. [1]. In their work, moving object detection is achieved using a three-level based background subtraction technique. At the pixel level they utilized a linear predictive Wierner filter to represent intensity variations of background pixels based on a recent history of L values

$$I_{S,t}(p) = -\sum_{k=1}^{L} a_k I_{S,t-k}(p), \tag{2.2}$$

where $I_{S,t}(p)$ is the predicted background model pixel intensity value for pixel p at frame t, $I_{S,t-k}(p)$ is a past value, and a_k are the prediction coefficients. For object detection any pixel value in the new frame $t+1$ that deviates significantly from the corresponding pixel values $I_{S,t}(p)$ of the background model are declared foreground.

The other two levels are region and frame levels. These higher levels address some of the problems, that can not be handled by the pixel level such as: sleeping person and light switch.

This technique can very slowly handle: moved object, time of day. However, it fails to address the sleeping person, shadow, waving trees, moved object, bootstrapping and light switch problems. Generally this technique is not suitable for real-time applications since it is computationally demanding and requires a lot of storage to handle the problem of moving foreground that corrupts the history values of the storage buffer.

2.1.4 Non-Parametric Model

In this technique, a kernel density estimation (KDE) based function is used to calculate a non-parametric estimate of the probability density function (*pdf*) of pixel values in the buffer. That is, the histogram of the pixel values in the buffer is used to calculate the parameters of the intensity/color distribution of each background pixel,

instead of estimating these parameters as it is the case in Gaussians-mixture based techniques. The work of Elgammal et al. [10], is an example of this technique, where they used a Gaussian kernel function $\eta(\mu_{t,k}, \Sigma_t)$

$$f(I_t(p)) = \frac{1}{L}\sum_{k=1}^{L}\eta(I_t(p); \mu_{k,t}, \Sigma_t) = \frac{1}{L}\sum_{k=1}^{L}\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_t|^{\frac{1}{2}}}e^{-0.5(I_t(p)-\mu_{k,t})^T\Sigma_t^{-1}(I_t(p)-\mu_{k,t})}, \tag{2.3}$$

where $f(I(p))$ is the *pdf* of pixel p intensity, $\eta(\mu_{t,k}, \Sigma_t)$ is the kernel Gaussian function, L in the order of 100-200, d is the model dimension (if RGB color space used then $d = 3$), the covariance matrix of color intensity values Σ_t is the same for all kernels (and assumed diagonal for simplicity)

$$\Sigma_t = \begin{pmatrix} \sigma_R^2 & 0 & 0 \\ 0 & \sigma_G^2 & 0 \\ 0 & 0 & \sigma_B^2 \end{pmatrix}. \tag{2.4}$$

For object detection, if $f(I_t(p))$ is smaller than some predefined threshold, then the current pixel is declared as foreground since it is unlikely to come from this distribution. Σ_t must be estimated, which is the key problem of the KDE technique given the limited length of the buffer (kernel window in time). In [10], the covariance matrix is estimated by median absolute deviation over the samples in the buffer for consecutive values of pixel location. Background model update is achieved by FIFO manner (the oldest sample is discarded and the newest sample is added to the model, and this new sample is chosen randomly from each interval of L frames) and given this new pixel value they have used two background models: Short term model and long term model with two mechanisms for updating the background pixel value: Selective update and blind update.

The presented technique [10] addresses time of the day and the sleeping person problems, with an overall objective of building a background model that adapts quickly to the changes in the scene and supports sensitive detection with minimum false negative/positive rates. However it fails to address the bootstrapping, light

switch, waving tree, moved object and shadow problems. The computational complexity and storage requirement is high and not suitable for real-time applications.

2.1.5 Other Less Promising Non-recursive Techniques

Various other techniques have been presented in the literature, that are not recursive in nature and require significant resources for initialization. In W4 [37,38], a temporal maximum and minimum inter-frame differences of all identified background pixels are calculated and stored, a pixel is marked foreground (object detection) if

$$|m - I_t(p)| > D \, or \, |N - I_t(p)| > D, \tag{2.5}$$

where $I_t(p)$ is pixel intensity, the per pixel parameters M, N and D represent the minimum, maximum and largest inter-frame absolute difference observed in the background scene. These parameters are calculated from the first few seconds (tens of seconds) of video and periodically updated selectively. W4 addresses the following canonical problems: time of day, sleeping person and noise related changes. It fails in addressing: moved object, waving trees, light switch, bootstrapping and shadows. W4 has high computational and storage requirement with low/medium accuracy.

In Eigenbackground [39], L sample frames of motionless backgrounds are collected, the mean and the covariance matrices are computed. Principle Component Analysis (PCA) is then used to compute the best eigenvectors and stored in a matrix ϕ_v. Then new incoming frames are projected onto the PCA subspace $I_{proj,t}(p)$. Since the eigenspace is a good model of the static parts of the scene but not for moving objects, differences $|I_t(p) - I_{proj,t}(p)| > T$ between the projection and the current frame greater than a threshold T are considered foreground. This technique can address: time of day, sleeping person and fails to address the critical problems of: moved object (waking person), waving trees, light switch, bootstrapping and shadows. The computational and memory requirement of the Eigenbackground is not high however the accuracy is also not high.

2.2 Recursive Techniques

This group of techniques is much more common compared to that of non-recursive. Recursive techniques do not require a storage buffer for background estimation, instead they recursively update the background models based on each new input frame. Most of these techniques include a weighing mechanism for limiting the errors due to noise and abrupt changes in the current input frame; they also incorporate positive decision feedback to use only those background pixels for updating.

2.2.1 Approximated Temporal Median Filter

This technique is based on a pixel-wise median filter approximation over time, which is a recursive alternative to the non-recursive median filtering that was discussed in Sec. 2.1.2. McFarlane et al. [40] and Remagnino et al. [41] used this technique for background modeling, in which the estimated pixel values of the current median image are incremented by one if the input pixel is larger than the corresponding pixel value in the estimated median image, or decreased by one if smaller. This estimate image eventually converges to a value corresponding to the temporal median. This technique requires at least one empty input frame without moving objects for good initial estimate of the median image (background model). The problems that can be addressed by this technique are: time of day and sleeping person, and those that can not be addressed are: the bootstrapping, moved object, waving trees, sudden illumination (light switch) and shadows. The approximated temporal median filter technique has the lowest computational and storage requirements, and it produces good results with an extremely simple implementation.

2.2.2 Kalman Filter and Single Gaussian

This is a common statistical technique with many different versions for background modeling. One of the simplest versions uses only the luminance intensity values, Karmann et al. [42] used both intensity values and their temporal derivatives, while Köller et al. [43] used both intensity and their spatial derivatives. The Pfinder [44]

uses a simple scheme of both luminance and color, and represents the background model by a single Gaussian distribution for each pixel

$$\eta(I_t(p);\mu_t,\Sigma) = \frac{e^{\left[-0.5(I_t(p)-\mu_t)^T\Sigma^{-1}(I_t(p)-\mu_t)\right]}}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}}, \tag{2.6}$$

where $\eta(I_t(p);\mu_t,\Sigma)$ is pixel intensity Gaussian distribution, $I_t(p)$ is pixel intensity, μ_t is the spatial mean, Σ is the covariance matrix of a per pixel Gaussian model that corresponding to the various regions of the background model. In each frame, only the mean values of visible background model pixel statistics are recursively and non-selectively updated by

$$\mu_t = \alpha I_t(p) + (1-\alpha)\mu_{t-1}, \tag{2.7}$$

where α is an adaptation constant.

Heikkila et al. [45, 46] used the same scheme as in Pfinder followed by a closing operation with a 3x3 kernel and small region discarding. Halevy et al. used a different background updating scheme as

$$\mu_t = \alpha S(I_t(p)) + (1-\alpha)\mu_{t-1}, \tag{2.8}$$

where $S(I_t(p))$ is a smoothed $I_t(p)$. The LOTS [47] system used three background models; primary, secondary and old background, and all updated selectively using the previous update formula of equation (2.7), after thresholding and getting the initial blobs a connected component is employed. A recent LOTS-based technique presented in [48], uses two background models, two threshold images I_{TL}, I_{TH} and Quasi Connected Component analysis. The background model and the lower threshold image I_{TL} are updated periodically every t_N. In ARMA [49] a new robust Kalman-filter based technique is presented to handle explicitly the non-stationary nature and clutter like dynamic background textures.

The previous statistical-based techniques aim to tolerate the background variations caused by noise, ambient illumination changes, and motion of non-stationary

objects, thus can address the following problems: time of day. Unless feedback or multiple background models are employed they fail to address: sleeping person, waving trees, bootstrapping, moved object, and shadows. The computational and storage requirements are high, and not suitable for real-time applications, the accuracy of these methods is medium as reported by various publications.

2.2.3 Mixture of Gaussians (MOG)

The MOG-based techniques are the most commonly used recursive techniques given their solid analytical and theoretical foundation. The first recursive version was presented in [3], where each pixel is modeled by a mixture of K Gaussians (components)

$$f(I_t(p)) = \sum_{k=1}^{K} \omega_{k,t}\eta(I_t(p); \mu_{k,t}, \Sigma_{k,t}), \tag{2.9}$$

where $f(I_t(p))$ is the *pdf* of pixel intensity, $I_t(p)$ is the intensity of pixel p at time instance t, $p = (x, y)$ is special pixel coordinates, $K = 3...5$ is the number of components, $\eta(I_t(p); \mu_{k,t}, \Sigma_{k,t})$ is the k-th multi-dimensional Gaussian component with intensity mean $\mu_{k,t}$ and independent covariance (standard deviation) $\Sigma_{k,t} = \sigma_{k,t}^2 E$ (E is the identity matrix) and $\omega_{k,t}$ is a weight corresponding to the priori probability of component k occurring. The Gaussian components for each pixel in the background model are updated before the foreground is detected (before thresholding) as follows:

i) if the new frame intensity data for a given pixel location $I_t(p)$ satisfies

$$|I_t(p) - \mu_{k,t}| / \sigma_{k,t} < 2.5, \tag{2.10}$$

then there is a match, and the k-th Gaussian component parameters are updated using an exponential decay scheme with an adaptation factor, as

$$\begin{aligned} \omega_{k,t} &= \omega_{k,t-1}, \\ \mu_{k,t} &= (1-\rho)\mu_{k,t-1} + \rho I_t(p), \\ \sigma_{k,t}^2 &= (1-\rho)\sigma_{k,t-1}^2 + \rho\left(I_t(p) - \mu_{k,t}\right)^T \left(I_t(p) - \mu_{k,t}\right), \end{aligned} \tag{2.11}$$

where $\rho = \alpha f\left(I_t(p) \mid \mu_{k,t-1}, \sigma^2_{k,t-1}\right)$, corresponds to the likelihood of that this pixel value was generated in state k. α is the adaptation rate coefficient (same as in single-Gaussian based techniques).

ii) Other Gaussian components that do not match, are updated according to

$$\begin{aligned} \omega_{k,t} &= (1-\alpha)\omega_{k,t-1}, \\ \mu_{k,t} &= \mu_{k,t-1}, \\ \sigma^2_{k,t} &= \sigma^2_{k,t-1}. \end{aligned} \tag{2.12}$$

iii) If $I_t(p)$ does not match any component, then the Gaussian component with the lowest $\frac{\omega_{k,t}}{\|\Sigma_{k,t}\|}$ is replaced with a new component which has $\mu_{k,t} = I_t(p)$, large $\Sigma_{k,t}$ and low $\omega_{k,t}$.

All Gaussian components for a given mixture are then sorted into the order of decreasing $\frac{\omega_{k,t}}{\|\Sigma_{k,t}\|}$, and the first B components (usually the first two) are considered as background models

$$B = \underset{b}{argmin}\left(\sum_{i=1}^{b} \omega_i > T_{bf}\right), \tag{2.13}$$

where T_{bf} is some predefined threshold. For object detection, if $I_t(p)$ does not match one of those B background components then this pixel at time t is considered as a foreground.

The recursive version of MOG as presented by Stauffer and Grimson [3], can effectively handle gradual illumination change, temporal clutter and sleeping person, but in addition to its computational complexity it has four important shortcomings of: bootstrapping, moved object, sudden illumination changes and shadows and ghosts. Various modified versions of the MOG have been presented in an attempt to address some of the previous shortcomings, [4] used two different sets of component parameter update formulas. The first used at the initial starting state incrementally decreasing in time, [4] then switch to recent update set of equations in an attempt to improve and fasten the adaptation issue of the MOG in scenes with initial high traffic (bootstrapping). [4] also presented a shadow detection scheme based on color infor-

mation. Other more complex schemes were presented to achieve better background model adaptation. In Zivkovic [6] additional negative weight $\alpha.c_T$ corresponding to Dirichlet prior is added to suppress the transient components that are not supported by input data (again α is an adaptation coefficient). In Lee [8], bootstrapping is handled by including a component match counter into the component parameter update, the author also presented a more complex component classification scheme and used winner-takes-all component matching to improve the overall computational performance. A more robust but very computationally demanding technique is presented by [13] in which gradient-based spatial information is used along with the temporal information to address both the initialization and moved object problems.

2.3 Summary

Non-recursive techniques are highly adaptive but may require significant storage and processing resources. The works of [23, 24], are the two most referenced techniques in literature of the non-recursive based BS techniques. Both techniques reported high background model adaptivity with good object detection accuracy.

Recursive techniques do not require a storage buffer for background modeling and adaptation, instead they recursively update the background models based on each new input frame data. MOG-based techniques are the most commonly used techniques of the recursive-based category. MOG-based techniques are highly adaptive and address many problems related to adaptive background modeling and object detection. The work of [8] is a recent representative MOG-based technique with good reported results.

We selected the methods [8, 23, 24] for implementation and comparison with the proposed techniques.

Chapter 3

Proposed Background Subtraction (Background Update and Object Detection)

In this chapter, we present in detail the proposed background update and object detection technique (Fig. 3.1). Video foreground object detection through simple subtraction with a fixed background model frame has the computational and storage advantage. However, this approach of non-adaptive model subtraction fails in uncontrolled outdoor environments of changing ambient illumination, and temporal clutter. Periodically updated adaptive background models can usually address these problems. Given the fact that background pixels have more temporal persistence and higher recurrence frequency than any foreground pixels, the most logical approach to maintain a background model is by statistical averaging of video frames. Recursive Gaussian mixture is commonly used cumulative-average-based update technique with solid analytical and theoretical foundations (see Chap. 2).

There are three key contributions of the proposed MOG-based technique compared to the techniques in [3,4,6,8,9]. In the background update stage they are: i) new faster component parameter adaptation formulas and ii) new selective matching scheme based on the combined approaches of component ordering and winner-takes-all. In the object detection stage it is: iii) new embedded hysteresis-based component matching

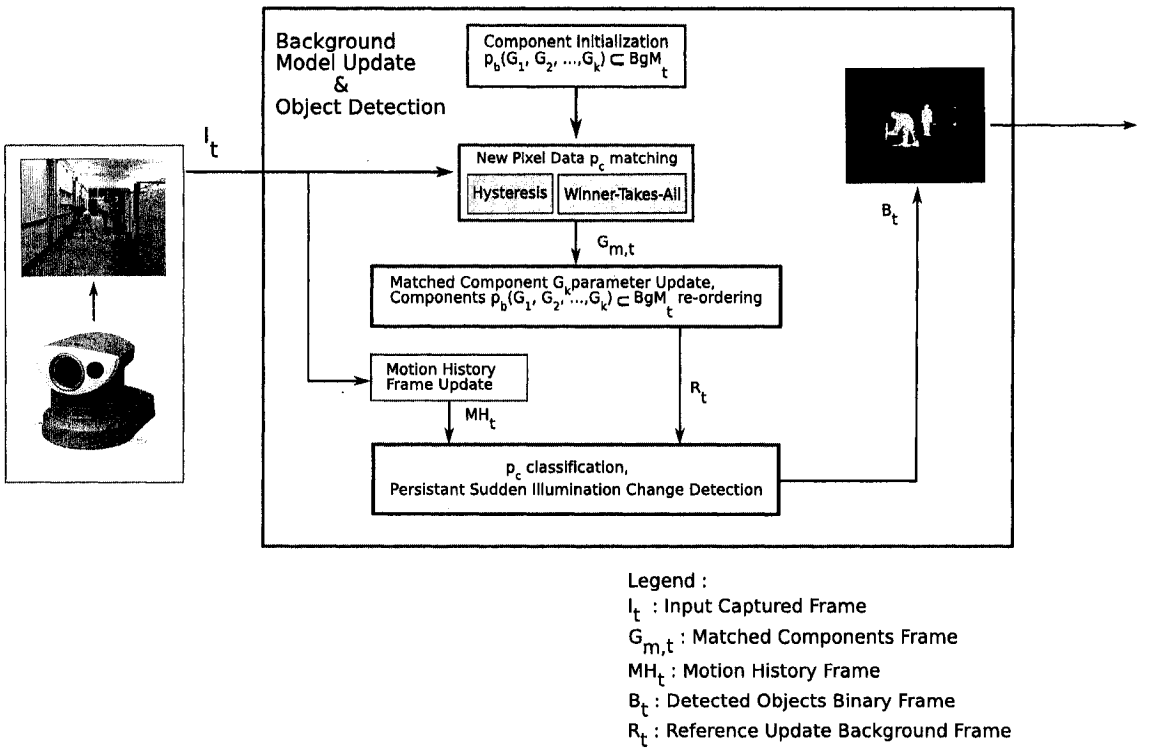


Figure 3.1: Block diagram of the proposed background update and object detection technique.

and temporal motion history schemes. The advantages of the proposed additions are improved performance and increased segmentation accuracy.

3.1 Notation

In the following, let

- I_t be the current input frame at time t,
- BgM_t the current background model,
- $p_c \in I_t$ is the input pixel at (i, j) location,
- $I_t(p_c)$ is the RGB or gray intensity value at p_c,
- $p_b \in BgM_t$ is the background model pixel at (i, j) location,
- $\{G_{1,t}, \ldots, G_{k,t}\}$ are the K Gaussian components of each p_b at time instance t,

- $\mu_{k,t}$ is the mean of the $G_{k,t}$ component,
- $\sigma_{k,t}$ is the $G_{k,t}$ variance, and assumed to be the same for all RGB color channels,
- $\omega_{k,t}$ is the weight or counter of $G_{k,t}$,
- $G_{m,t}$ is the matched component at time t,
- λ_1 and λ_2 are the hysteresis-based variance thresholds,
- ω_{gts} is frame counter (pixel data integration threshold),
- α is component parameter adaptation rate (i.e., controls how fast the mean, variance and weight of a $G_{m,t}$are adapted),
- ω_{norm} is the normalization weight threshold,
- T_{stop} is the sleeping person time threshold,
- λ_{bf} is secondary background component threshold,
- n_{sdn} is the number of p_c that are classified as foreground pixels per new frame,
- T_{sdn} is the sudden illumination threshold,
- MH_t is the current motion history frame that is the accumulated result of subsequent temporal differencing,
- $p_{mh,t} \in MH_t$ is the motion history of a pixel at (i, j) location,
- h_L is the history length,
- $ps_{proc} \in BgM_t$ is surrounding processed-pixels of p_c within a defined area,
- F_{size} is current frame size,
- B_t is the detected object frame (foreground mask).

3.2 Model Formation

Each pixel of a video frame can be represented as a continuous random variable X. X can be one-dimensional (gray-scale values) or three dimension $3D$ (RGB or YUV values). At any given time t each pixel may correspond to one of different foreground or background object surfaces $k \in \{1, 2, \ldots, K\}$, where $K \subseteq [3-7]$ is the possible number of surfaces that come into the view of a given pixel. Some of the K states correspond to background surfaces, the rest are deemed to be foreground objects. Based on this assumption the foreground object detection problem can be solved by estimating which of the k surfaces gave rise (cause) the current pixel sample $x \subset X$. This is a classical pattern classification problem of finding the maximum posterior probability $P_t(k|x)$ for the current X reading, formulated by Bayes's theorem [50] as following

$$P_t(k|x) = \frac{P_t(x|k)\ P_t(k)}{P_t(x)}, \tag{3.1}$$

where $P_t(x|k)$ is the likelihood probability of x corresponding to k, $P_t(k)$ is the prior probability of a surface k, $P_t(x) = \Sigma_{k=1}^{K} P_t(k|x)P_t(k)$ is the evidence that is merely a scale factor.

Thus

$$max\left[P_t(x|k)\right] = \underset{k}{argmax}\left[P_t(x|k)\ P_t(k)\right]. \tag{3.2}$$

In order to use equation (3.2) to find the maximum matching surface k, both distributions of $P_t(x|k)$ and $P_t(k)$ must be defined in order to compute the corresponding probabilities. The probabilistic distribution of X given a surface k can be model as Gaussian

$$P_t(x|k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_{k,t}|^{\frac{1}{2}}} e^{-0.5(x-\mu_{k,t})^T \Sigma_k^{-1}(x-\mu_{k,t})}, \tag{3.3}$$

where $\mu_{k,t}$, $\Sigma_{k,t}$ are the mean and covariance of $P_t(x|k)$ at time t, D is the dimension of X, and T indicates matrix transpose. The $\mu_{k,t}$, $\Sigma_{k,t}$ parameters of each $P_t(x|k)$ are unknown but can be recursively estimated from new pixel data, as described in Sec. 3.3.2.

The prior probability $P_t(k)$ can be described in a classical statistical way as a

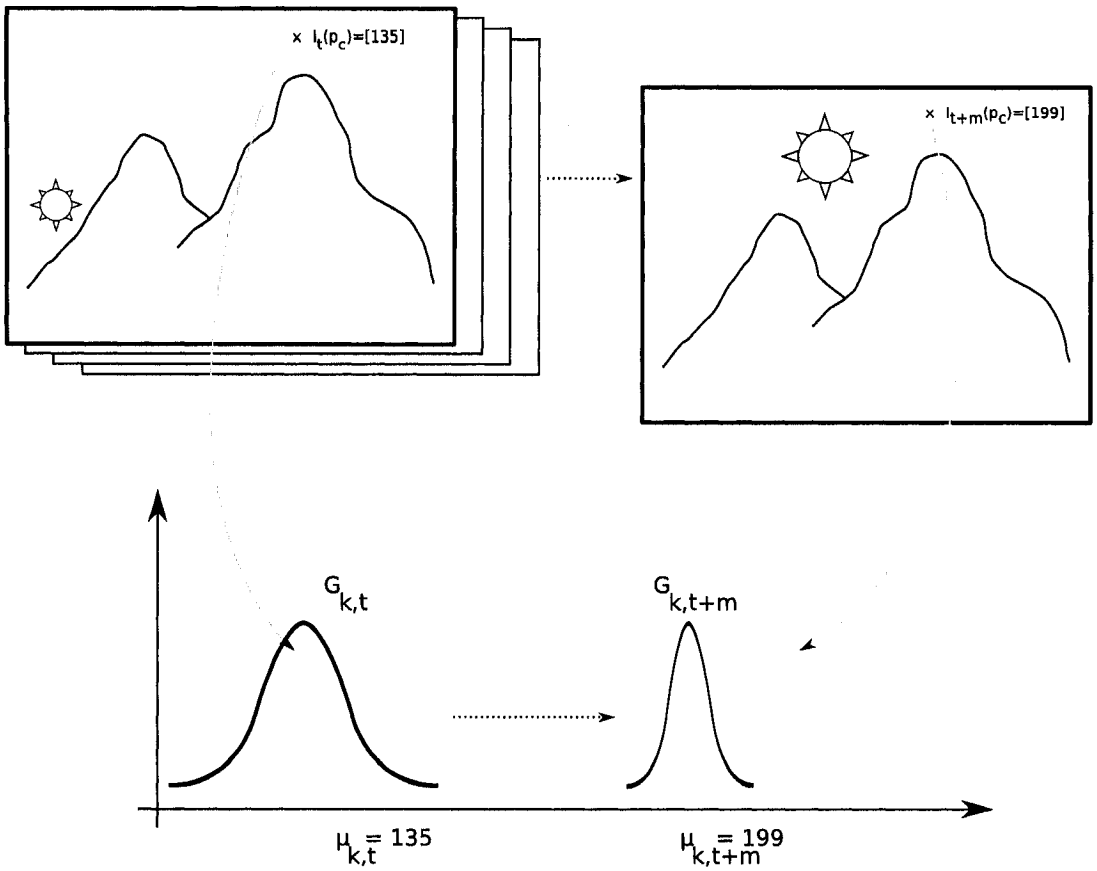


Figure 3.2: Component tracking of pixel intensity in time.

weighing factor that reflect the number of matches (occurrences) for a given surface k. After finding the matching surface k, k must be classified either as a foreground or a background.

From above, a given background model BgM_t corresponds to a single adaptive frame in which each pixel consists of k Gaussian components $G_{k,t}$. Components $G_{k,t}$ are nothing but different cumulative clusters of RGB or pixel intensity variations. Component means and variances are recursively updated. These different components track changing background and foreground intensity variations, only adapting to the most persistent in a predefined time slice (Fig. 3.2). New pixel data are compared and classified according to the current accumulated $G_{k,t}$.

The proposed technique consists of two stages, background model update and object detection. Initially all the components $\{G_{k,t}\}$ are reset as following: $\mu_{k,t} = \mu_{init} \;:\; \mu_{init} \gg 255$, $\sigma_{k,t} = \sigma_{init}$, $w_{k,t} = 0$, $MH_t = 0$ corresponding to no motion,

$w_{gts} = 0$ corresponding to integrate all new pixel data instantly.

3.3 Background Update

In this stage, the parameters of all $G_{k,t}$ are adapted according to the new frame pixel data. Note that $\mu_{1,t}$ and $\mu_{2,t}$ (if $G_{2,t}$ have enough supporting statistics) of each pixel p_b correspond to the updated background model frame $BgM_{1,t}$ (or frames $BgM_{1,t}$ and $BgM_{2,t}$) used in BS. This stage consists of: selective component matching and component parameter updating.

3.3.1 Selective Component Matching

Selective matching (comparison) of new pixel data $I_t(p_c)$ is achieved as following: i) components G_k are always descendingly ordered according to their weights $w_{k,t}$, thus matching always starts first with the most probable component G_1, ii) upon the first match where

$$|I_t(p_c) - \mu_{k,t}| < \lambda_1 \sigma_{k,t}. \tag{3.4}$$

Matching terminates (winner-takes-all) and matched component's parameters $\{\mu_{k,t}, \sigma_{k,t}, w_{k,t}\}$ are updated, iii) if no match occurs, parameters of component $G_{k=K}$ with smallest weight are reset $\mu_{K,t} = I_t(p_c)$, $\sigma_{K,t} = \sigma_{init}$, $w_{K,t} = 1$. For computational reasons, hysteresis is only used if a match occurs with the first component G_1 (BgM_t component), then G_1 parameters are updated with the new pixel data only if

$$\lambda_2 \sigma_{1,t} > |I_t(p_c) - \mu_{1,t}| \vee \exists ps_{proc} \notin foreground. \tag{3.5}$$

The proposed component re-ordering and selective winner-takes-all matching schemes significantly improve performance and prevent possible component overlap (Fig. 3.3). The proposed hysteresis component matching scheme improves detected foreground object blobs by reducing the amount of cracks and shadows.

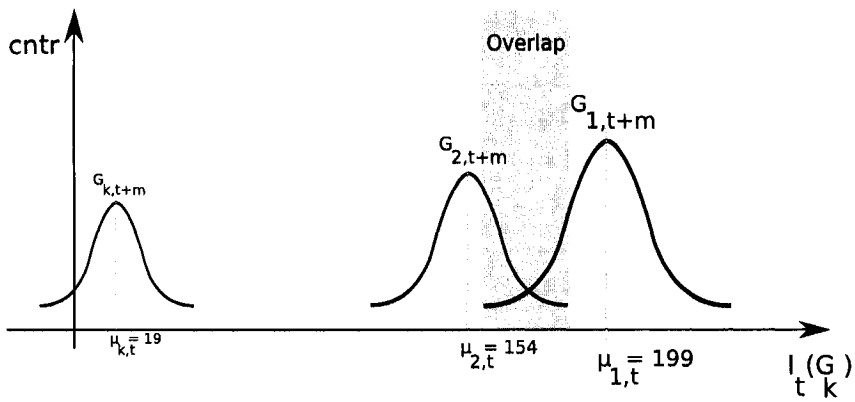


Figure 3.3: $G_{k,t}$ components overlap for a given pixel p_c.

3.3.2 Component Parameter Update

Matched component's parameters are updated as

$$\begin{aligned} \mu_{k,t} &= \mu_{k,(t-1)} + \alpha.\left[\left|I_t(p_c) - \mu_{k,(t-1)}\right|\right], \\ \sigma_{k,t} &= \sigma_{k,(t-1)} + \alpha.\left[\left|I_t(p_c) - \mu_{k,t}\right| - \sigma_{k,t-1}\right], \\ w_{k,t} &= w_{k,t-1} + 1. \end{aligned} \tag{3.6}$$

However, in case persistent sudden illumination change is detected (Sec. 3.4.3), then set $w_{k,t} = w_{1,t}$.

Following parameters update, a periodic (i.e., not every frame) weight normalization step is performed. First all components $G_{k,t}$ are descendingly re-ordered. All components weights are normalized periodically if $w_{gts} > w_{norm}$, such to make the largest weight $p_b(w_{1,t}) > 2T_{stop}$ and then reset $w_{gts} = 0$. This insures fast and correct bootstrapping recovery, and prevents temporary stopped object (sleeping person) pixels from integrating into the background model.

3.4 Object Detection

In this stage, new frame I_t pixels p_c are grouped into foreground or background. This stage has the following steps: motion history, pixel classification and sudden illumination change detection.

3.4.1 Motion History

The proposed motion history scheme helps reduce cracks and enhance real moving object boundaries. Motion history at each pixel is calculated as

$$p_{mh,t} = \begin{cases} h_L & : \quad |I_t - I_{t-1}| > T_{TD} \\ p_{mh,t-1} - 1 & : \quad |I_t - I_{t-1}| \leq T_{TD} \wedge p_{mh,t-1} \neq 0 \end{cases}, \tag{3.7}$$

T_{TD} corresponds to the per pixel temporal difference threshold between consecutive frames. In the motion history computation, we used two consecutive frame history $h_L = 2$ and a temporal difference threshold $T_{TD} = \lambda_1 \sigma_{k,t}$.

3.4.2 Pixel Classification

Each pixel p_c is classified as background only if the matched component was $G_{1,t}$, or the weight of the matched component satisfies $w_{k,t} > \lambda_{bf} w_{1,t}$ (have enough supporting statistics). However, if $p_{mh,t} > 0$ then p_c corresponds to foreground even if it matched $G_{1,t}$, or $G_{k,t}$with a weight $w_{k,t} > \lambda_{bf} w_{1,t}$. This insures the blob-boundary-integrity of real moving objects in the foreground mask.

Another possible classification approach is to use the updated background model frames $BgM_{1,t}$, $BgM_{2,t}$ and the adaptive thresholding technique of [51] in the BS.

3.4.3 Sudden Illumination Detection

After classifying all pixels in I_t, the frame counter w_{gts} is increment by one. Persistent sudden illumination changes can happen due to cloud cast or sudden global light projection. Detection of sudden illumination change is accomplished by counting the number of $p_c(t)$ that are classified as foreground pixels (n_{sdn}). If $n_{sdn} > T_{sdn}$ and the change persists for more than predefined number of frames, a sudden illumination change event is determined.

3.5 System Implementation and Parameters

The proposed background update technique was incorporated into a real-time outdoor monitoring system. The system's input video stream was obtained from multiple fixed visual sensors at a constant rate of 30 frames per second (fps). We also tested the system on gray-scale and color (RGB) test video sequences of real traffic scenes. In the case of gray-scale streams and sequences, the proposed model implementation is straight forward. For color RGB streams and sequences, we use the red, green, and blue channels for $G_{k,t}$ components matching and updating, and only the intensity $Y = R + G + B$ for motion history. This means that $G_{k,t}$ components $\mu_{k,t}$ consists of $\mu^r_{k,t}$, $\mu^g_{k,t}$ and $\mu^b_{k,t}$ for the red, green and blue channels respectively.

We chose the number of components per pixel $K = 4$. For parameter initialization we used $\mu_{init} = 999$, $\sigma_{init} = 10.0$, $\alpha = 0.005$. The choice of hysteresis thresholds λ_1, λ_2 is dependent on the monitored scene. For scenes with small foreground moving objects, typically when objects and their cast shadows has size less than 10% of F_{size}, it is good to use $\lambda_2 \in [1.0 - 1.5]$ and $\lambda_1 \in [1.5 - 2.0]$. Note that λ_2 corresponds to the base threshold that controls the minimum intensity difference required for being a background pixel. While λ_1 generally controls the amount of small cracks and added shadows in/to foreground blobs. In the case of large foreground objects, with a size larger than 20% of F_{size}, it is good to use $\lambda_2 \in [2.0 - 3.0]$ and $\lambda_1 \in [3.0 - 5.0]$.

Weight normalization thresholds are chosen as following, assuming that the monitored scene has a traffic light or inter-section where vehicles stop temporarily for up to a minute. That means some foreground objects frequently may become static for $50\,seconds$, corresponding to $T_{stop} = 50 \times 30\,fps = 1500\,frames$, then the normalization threshold must be $w_{norm} > \frac{2T_{stop}}{(\lambda_{bf}+0.5)}$ where 0.5 is error margin corresponding to foreground object stop time tolerance. To normalize we simply divide all component weights by two. The choice of sudden illumination threshold T_{sdn} depends on the scene and visual sensor frame size, assuming that the biggest moving foreground object (e.g., of a bus or a truck) takes up to 35% of F_{size}, we chose $T_{sdn} = 0.75\,F_{size}$.

In component matching and for color streams or sequences, the conditions in

Eqs. (3.4, 3.5) are implemented to correspond for the red, green and blue channels respectively as

$$max\left[\left|I_t^r(p_c)-\mu_{k,t}^r\right|,\left|I_t^g(p_c)-\mu_{k,t}^g\right|,\left|I_t^b(p_c)-\mu_{k,t}^b\right|\right]<\lambda_1\sigma_{k,t},$$
$$\lambda_2\sigma_{1,t}>max\left[\left|I_t^r(p_c)-\mu_{k,t}^r\right|,\left|I_t^g(p_c)-\mu_{k,t}^g\right|,\left|I_t^b(p_c)-\mu_{k,t}^b\right|\right]\vee\exists ps_{proc}\notin foreground. \tag{3.8}$$

Component updating (Eq. 3.6) are done in a similar way for color sequences

$$\mu_{k,t}^r=\mu_{k,t-1}^r+\alpha\left[I_t^r(p_c)-\mu_{k,t-1}^r\right], \tag{3.9}$$

$$\mu_{k,t}^g=\mu_{k,t-1}^g+\alpha\left[I_t^g(p_c)-\mu_{k,t-1}^g\right], \tag{3.10}$$

$$\mu_{k,t}^b=\mu_{k,t-1}^b+\alpha\left[I_t^b(p_c)-\mu_{k,t-1}^b\right], \tag{3.11}$$

$$\sigma_{k,t}=\sigma_{k,t-1}+\alpha.\left[max\left|I_t^{r,g,b}(p_c)-\mu_{k,t}^{r,b,g}\right|-\sigma_{k,t-1}\right]. \tag{3.12}$$

In object detection stage, the secondary background component threshold λ_{bf} depends on the monitored scene and its value must be carefully considered. Depending on the clutter frequency of the non-stationary dynamic pixels of the background in the monitored scene, λ_{bf} can have the following range of values $\lambda_{bf}=[0.3-0.9]$.

3.6 Analysis and Comparison to MOG Methods

We present here the functional analysis of the proposed background subtraction technique and compare it to the MOG-based techniques in [3–6, 8, 9, 13, 52].

3.6.1 Contributions Analysis

i) The proposed novel hysteresis component matching scheme (Sec. 3.3.1) significantly improves segmentation accuracy with minimum added computational overhead. None of the reviewed techniques incorporated a similar scheme, instead most of the proposed techniques in the literature rely on post processing steps of either connected components or dilation followed by erosion, in order to reduce the small gaps in fore-

ground detected objects and to remove the salt and pepper like noise. However, these steps can not address largely dis-assembled foreground blobs (false negative) or the presence of large noisy blobs (false positive), nor it can reduce the affect of shadows and ghosts. In addition, most of the time these schemes significantly reduce the accuracy of the detected foreground objects especially at their boundaries which can be important for some applications. The hysteresis thresholds λ_1, λ_2 can be viewed as two low pass filters, where their cut-off frequencies control the amount of pixels that are classified as foreground. The choice of hysteresis thresholds is application dependent and directly affects the integrity of foreground blobs (false negative), the amount of added shadows and ghosts (false positive), and the scattered noise (false positive).

ii) The proposed component mean update equations (Eqs. 3.6, 3.9, 3.10, 3.11) (Sec. 3.3) are reformulated to improve calculation performance. The simple component variance update equation (Eqs. 3.6, 3.12) is novel formula that greatly improves performance with no adverse affects as we found from our simulation. The importance of these update equations not only to accurately estimate the parameters, but also to accommodate for some non-stationarity in pixel values $I_t(p_c)$. This allow the parameters of $G_{k,t}$ to adapt to changing ambient illumination by emphasizing more recent samples over older ones.

iii) The proposed component weight update technique uses a simple incremental scheme (Sec. 3.3). This approach not only improves weight calculation performance but also insures fast initial model adaptation and implicitly addresses the bootstrapping problem (this problem was first indicated by [4]). The proposed periodical weight normalization is important for addressing the critical problem of temporary sleeping person (e.g., when traffic light stopped vehicles) that may stay for long time (up to two minutes $120 \times 30 = 3600$ frames) and should not be merged into the background components.

iv) The proposed motion history (Sec. 3.4.1)enhances moving foreground objects boundaries with no adverse affect. This added boundary is important for reducing the gaps in moving object boundaries without the need for a separate component

connected processing as it is the case for most MOG-based techniques [3–6, 8, 9, 53]. The effect of added motion history functionality is most evident in sequences where the moving foreground object's size is larger than 20% of F_{size}(indoor or confined monitored areas sequences). For sequences with small moving foreground objects, motion history has no effect (the output contribution is negligible).

3.6.2 Addressed Problems

The proposed technique, as the case for all recursive MOG-framework based BS techniques, can implicitly handle a realistic environment's: gradual illumination change, sleeping person, and temporal clutter (waving trees). However, the first presented MOG framework of Stauffer and Grimson [3] fails to address four problems: bootstrapping, moved object, sudden illumination changes, and cast shadows and ghost. Following presented MOG-based BS techniques attempted to address these four shortcomings.

i) The proposed component weight update scheme addresses the bootstrapping problem as shown in Sec. 5.6.

ii) The moved object problem [1] is not addressed by most of the presented MOG-based techniques including the proposed. However, the proposed technique addresses a closely related problem of sleeping person where moving object temporally become stationary, as in the traffic light stopped vehicles where the stationarity time can vary from one to two minutes corresponding to $1800 - 3600$ frames. The proposed periodical weight normalization prevents pixel components G_k that correspond to those stationary foreground objects from becoming dominant, and thus from getting partially or fully merged into the background model. In most MOG-based techniques, the chosen adaptation rates were $\alpha \in [0.001 - 0.05]$ corresponding to $20 - 1000$ frames. This leads most of the time to either partial or full merger of these stationary foreground objects into the background model (false negative) as shown in Fig. 3.4-b. Then when these objects start to move again the problem of moved objects occurs leaving ghost in the foreground mask (false positive) as shown in Fig. 3.4-e. This cycle keeps repeating and continuously producing false segmentation. A pragmatic but

computationally expensive solution can be achieved by incorporating higher level logical processing steps or by using spatial edge information as presented by [13], where any false detected ghost in the temporal scheme that has no significant supporting data in the spacial scheme are disregarded.

iii) The problem of sudden global illumination change is not addressed by most presented MOG framework techniques. This problem occurs very frequently in realistic environments mainly due to cloud cast with varying pattern of occurrences. The proposed sudden illumination change is very simple yet effective, a similar approach was adopted by [13]. The main differences with the proposed technique is that we verify change persistent before setting the corresponding components as a secondary background components. The reason for this delay is to avoid the transient global and local sudden illumination changes that can pass T_{sdn} threshold. These transient changes can happen due to e.g., small shallow clouds, spot light projection, and foreground aperture problem [1].

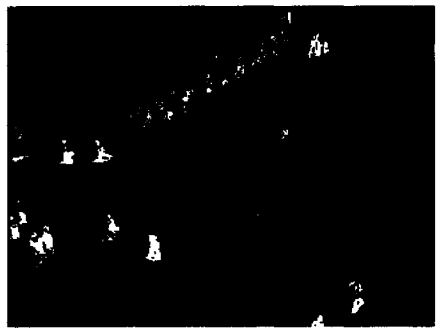
iv) The problem of cast shadows and ghosts is common key problem to all video segmentation techniques. Most of the reviewed MOG-framework based techniques try to provide additional processing steps to partially address this problem. In our case we rely on the proposed hysteresis thresholding scheme to reduce the effect of shadows and ghost (false positive) while retaining the integrity of actual foreground blobs (true positive) (Fig. 3.5).

3.6.3 Computational Performance Analysis

Computational complexity of any recursive MOG-based technique is directly related to its model formation.The most significant contributions of the proposed formation to the state of the art MOG framework that directly related to computational performance are:

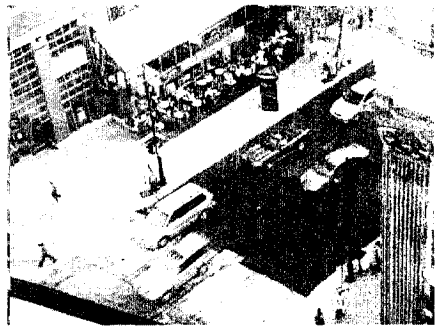
i) Descendingly re-ordering components G_k according to their weights combined with the winner-takes-all mechanism not only allows matching the new data with the most probable (dominant) components first, greatly improving performance, but also simplifies pixel classification and component replacement in case of no match. Most of

(a) Original St.Catherine-St.Guy crossing, I_{7600}.

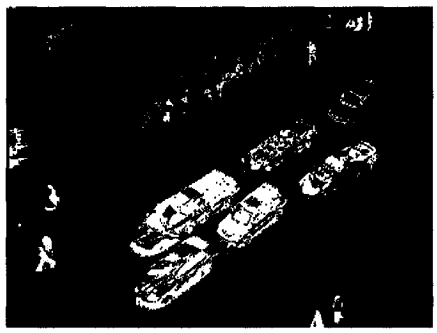


(b) Output of [8]. Detected objects are merged into the background.

(c) Ouput of the proposed technique. Foreground objects are correctly detected and not merged into the background.



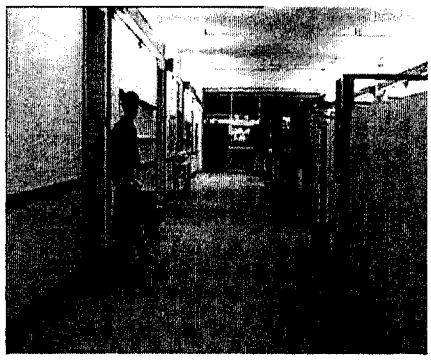
(d) I_{7708}.



(e) Output [8]. False ghosts corresponding to the moved vehicles are detected.

(f) Output of the proposed technique. Correct detection with no ghost.

Figure 3.4: The dual problem of merged foreground objects and the subsequent false ghosts in most MOG techniques. In (b) all temporary stopped vehicles are wrongly merged into the background model. For (c) the proposed technique correctly detected temporary stopped vehicles without merging them into the background model. In (e) all the detected vehicle objects actually correspond to false ghosts of these (once merged) moved vehicles. For (f) the proposed technique correctly detected objects with no false ghosts.

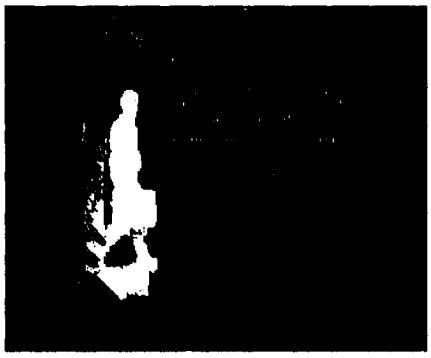


(a) Hall original I_{30}.

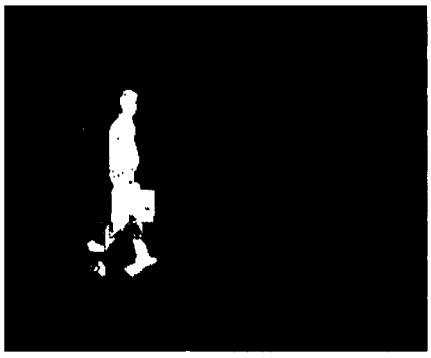
(b) Proposed technique output, $\lambda_1 = 3$, $\lambda_2 = 2$.

(c) Proposed technique output, $\lambda_1 = 4$, $\lambda_2 = 2$.

(d) Proposed technique output, $\lambda_1 = 4$, $\lambda_2 = 3$.



(e) Proposed technique output, $\lambda_1 = 4$, $\lambda_2 = 1$.



(f) Proposed technique output, $\lambda_1 = 5$, $\lambda_2 = 3$.

Figure 3.5: Shadow and ghost reduction using the proposed hysteresis thresholding scheme. All parameters are fixed except for λ_1, λ_2.

the other reviewed recursive techniques tend to either match new pixel data with all the components and then re-order them (or their weights) prior to pixel classification or components replacement in case of no match.

ii) New simple variance update formula Eq. 3.12 that improves performance. The new formula always conveyed the important concept of emphasizing the effect of more recent data samples over the older ones. Other reviewed techniques propose a more computationally demanding update formulation similar to that of $\sigma_{i,t}^2 = (1-\rho)\sigma_{i,t-1}^2 + \rho\left(I_t(p) - \mu_{i,t}\right)^T \left(I_t(p) - \mu_{i,t}\right)$ as originally presented by [3].

iii) The proposed hysteresis matching scheme introduces added computational load that depends on the chosen neighborhood comparison window and the application (processed scene). The motion history is very simple to implement thus its computational load is negligible. The storage requirement of the proposed technique aside from the motion history buffer is same as other MOG techniques.

3.7 Analysis and Comparison to Non-MOG Techniques

In this section, we present the functional analysis for two commonly-referenced non-MOG techniques and compare them analytically to the proposed technique. These techniques are: i) median-filter based technique of [23] and the KDE non-parametric technique of [10, 14, 24].

i) The pixel level median-filter based technique [23] can adapt to gradual illumination changes, but fails to address various problems without the higher level logical stages. Most of the problems are related to the limited length of the buffer L, since capturing the real-background statistics requires that more than half of the buffer should correspond to background samples. A key problem in [23] is that all the samples from foreground moving objects with different color have the same weight in the buffer, i.e., samples from a red color moving foreground object has the same contribution in the buffer as that of a blue object, unlike the MOG framework, where

each different foreground object with different colors is treated as a separate component. This key problem affects directly the adaptation rate of the median-based technique, making it slower than the MOG-based techniques in addressing the bootstrapping, moved object and sleeping person problems. Other problems of temporal clutter, shadows and ghost can not be handled even when a long buffer is used. The computational complexity of computing the median is higher than that of the MOG framework for large buffers $L > 80$, even when more optimized functions are used [54]. The storage requirement of this technique is higher than MOG-based techniques.

ii) The work in [10] addresses most of the problems that MOG-based technique can handle. However, temporal clutter can only be address given that cluttering pixel displacement is limit to within neighboring window of five pixels. The computational and storage requirement of this technique is very high even with efficient implementation using precalculated lookup tables for the kernel function values and by partial evaluation of the sum in $f_{pdf}(I_t(p)) = \frac{1}{L}\sum_{i=1}^{L}\prod_{j=1}^{d} K_{\sigma_j}\left(I_{j,t}(p) - I_{i,j,t}(p(i)]\right)$(Eq. (7) in [14]), where $d = 3$ is a dimensional vector representing the color. Also, and since it requires a more complex post processing step than connect components [3] for handling noise and filling gaps and removing temporal clutter false detection, it is much more computationally complex than any MOG-based technique.

3.8 Summary

We proposed a novel MOG-based real-time video background update technique for moving object detection. The background update stage consists of a new combined approach of component ordering and winner-takes-all. This matching scheme not only selects the most probable component for the first matching with new pixel data, greatly improving performance, but also simplifies pixel classification and component replacement in case of no match. Further performance improvement achieved by using a new simpler yet functional component variance adaptation formula.

In the object detection stage of this technique, the proposed new hysteresis based component matching and temporal motion history schemes improved segmentation

quality. Component hysteresis matching improves detected foreground object blobs by reducing the amount of cracks and added shadows, while motion history preserves the integrity of moving objects boundaries, both with minimum computational overhead. The proposed periodical weight normalization scheme prevents merging temporary stopped real foreground object, and the creation of false ghosts in the foreground mask when these objects start to move again.

The proposed overall technique implicitly handles both gradual illumination change and temporal clutter problems. The problem of shadows and ghost is partially address by the proposed hysteresis based matching scheme. The problem of persistent sudden illumination changes and camera movements are addressed at frame level depending on the percentage of pixels classified as foreground.

The algorithm analysis shows that the proposed background update and object detection technique not only robustly addresses more background adaptation related problems compared to other MOG and non-MOG based techniques, but also computationally more efficient as we objectively show in the following chapter 4.

Chapter 4

Experimental Results of Proposed Background Subtraction

Our goal in this chapter is to evaluate the core performance of each algorithm at pixel level without the added extra pre-processing or post-processing steps. In the implementation of [23], higher processing stages of object tracking and logical correspondence are not included since they are beyond the scope of this thesis. In the implementations of [14,24], [8], we did not include the data validation and /or shadow removal steps.

We selected three especially challenging (publicly available) color urban traffic sequences (form the website maintained by KOGS/IAKS of Karlsruhe university http://i21www.ira.uka.de/image_sequences/). The first sequence "Winter" or "dtneu_winter" is 300 frames of size 768x576 and shows a traffic intersection in snowing winter day-light, it contains pedestrians and moving vehicles. The sequence starts with moving pedestrians and vehicles (bootstrapping), but no stopped and later moved foreground objects. The other two sequences "Fog" or "dtneu_nebel" (336 frames) and "Snowing" or "dtneu_schnee" (300 frames), show the same intersection under different weather conditions and heavier traffic activity.

In the development process of the proposed BS technique, we conducted long hours of real-time simulations on video streams obtained from network cameras for different scenes. In order to evaluate the long term performance of each algorithm,

we used our own captured long real traffic video sequences in downtown Montreal. We present the simulation results for two such sequences, the first is 2850 frames, it shows the intersection of Rene-Levesque and Guy at mid-day time, with slight rain, waving trees, and normal downtown traffic. The second is 1200 frames and shows St-Catherine street in the afternoon, with slight wind, clouds and normal downtown vehicle and pedestrian traffic.

We present first quantitative objective evaluation of algorithm performance for the "Hall" and "Intelligentroom" sequences given that we have the ground-truths. These are two well known test sequences, "Hall" is 300 frames of size 352x288 of and "Intelligentroom" is 300 frames of size 320x240. For the remaining test sequences, objective evaluation was not feasible as it requires unavailable ground-truth sequences. For these test sequences we provide only subjective object-detection performance evaluation approach that is based on i) the integrity and correct detection of real moving foreground objects (true positive and false negative) and ii) the amount of false detection of noise, shadows and ghosts (false positive).

Note that in both objective and subjective performance evaluations, and for all test sequences, we used default parameter values of each reference technique as stated in the corresponding publications [8, 14, 23]. For the proposed technique we used the following parameter values $\alpha = 0.005$, $\lambda_2 = 2.0$, $\lambda_1 = 3.0$.

Algorithm simulation were achieved via our multi-threaded C++ based application. The simulation platform is a Linux (FC4) based Intel P4 @ 2.0 Ghz with 512 MB of system memory.

4.1 Objective Evaluation

We objectively quantify object detection performance of each algorithm at the pixel level against the object detection ground-truth of the "Hall" and "Intelligentroom" sequences, by means of Euclidean distance [50] defined as

$$d^2_{E_F}(O, O_{truth}) = \sum_{k}^{M} \sum_{l}^{N} (O(k,l) - O_{truth}(k,l))^2, \tag{4.1}$$

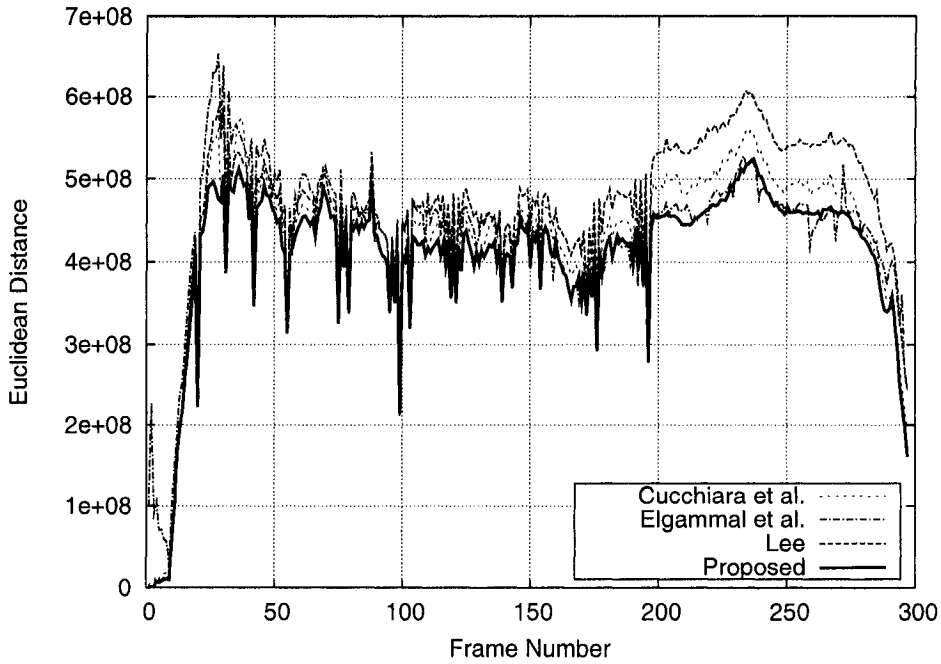


Figure 4.1: Euclidean distance objective comparison for "Hall" sequence.

where M and N are the width and height of a frame, and for a given BS technique, $O(k,l)$ is the algorithm output at location (k,l), and $O_{truth}(k,l)$ is the objects ground-truth at location (k,l). Note that lower values for this measure corresponds to more correct object detection.

For "Hall" sequence, the output of the proposed technique better matches the ground-truth corresponding to more accurate object detection compared to other reference techniques [8,14,23] (Fig. 4.1). For "Intelligentroom" sequence, Fig. 4.2 shows more clearly the advantage of accurate object detection by the proposed technique over the reference techniques [8,14,23].

4.2 Subjective Evaluation

4.2.1 "Fog" Sequence

In the subjective evaluation of this sequence, we note the following:

i) Both the proposed and the median based [23] techniques, were the fastest to recover from bootstrapping problem given that the sequence starts with moving fore-

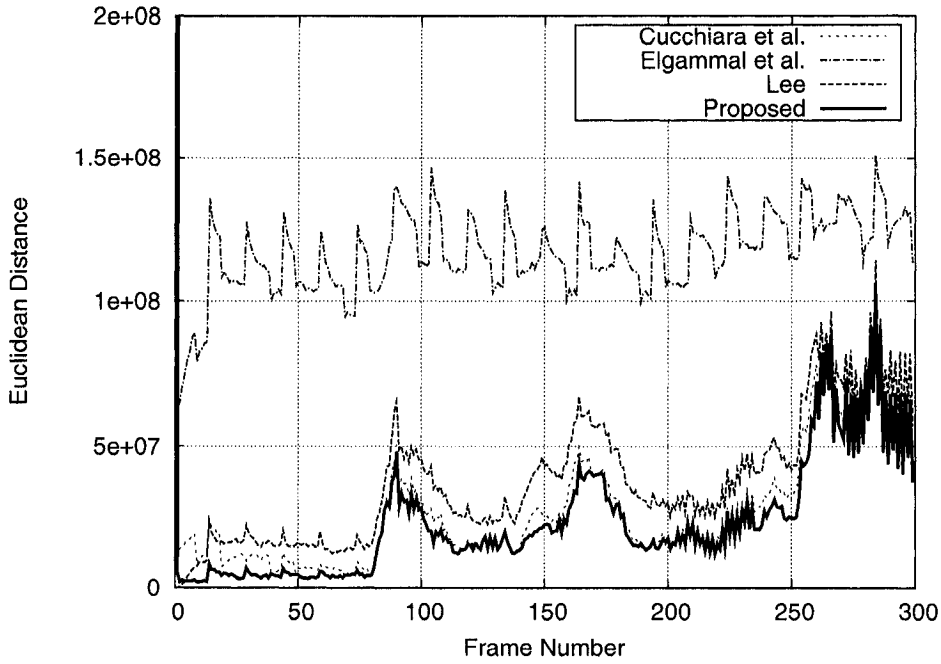
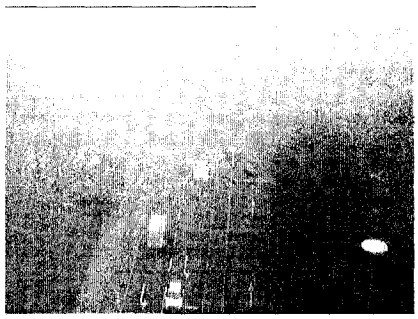


Figure 4.2: Euclidean distance objective comparison for "Intelligentroom" sequence.

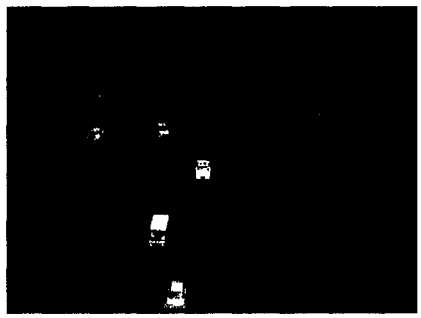
ground objects. The KDE-based technique of [14, 24], shows motion history (ghost trace) of the moved objects. This happened due to the ragged density estimate obtained of the very few sample set given the periodical model update of $\Delta t = 10$ as illustrated in Fig. 4.3. A more visible trace of initial object ghosts was present in the output of [8] technique mainly due to slow adaptation rate.

ii) In almost all the frames, the most correct object segmentation was obtained by the proposed technique in terms of foreground blobs integrity and the noise present (Figs. 4.3,4.4).

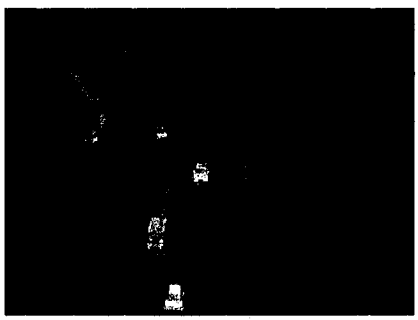
iii) All the algorithms, including the proposed, for some frames, failed to detected the three moving dark color vehicles due to heavy fog presence and color resemblance with the auto-route. However, in various frames there was partial object detection by the proposed technique mainly due to the proposed motion history and hysteresis thresholding schemes (Fig. 4.4). This indicate the better performance of the proposed technique.



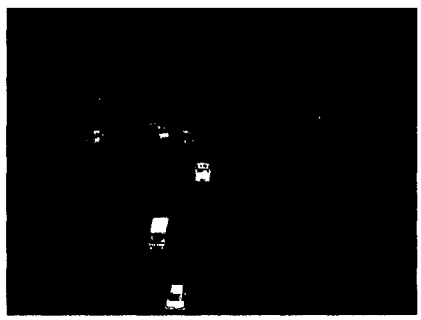
(a) Original "Fog", I_{110}.



(b) Output of [23].



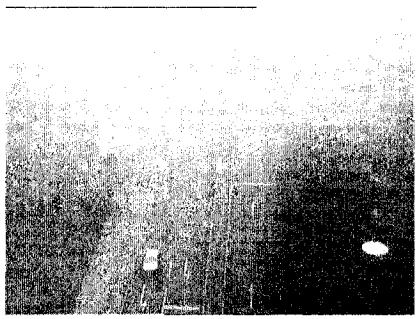
(c) Output of [24].



(d) Output of [8].

(e) Output of proposed technique.

Figure 4.3: Algorithm outputs for sample frame I_{110} of "Fog" sequence. The proposed technique output has more filled regions, boundaries and less noise.



(a) Original "Fog", I_{210}.

(b) Output of [23].

(c) Output of [24].

(d) Output of [8].

(e) Output of proposed technique.

Figure 4.4: Algorithm outputs for sample frame I_{210} of "Fog" sequence. The proposed technique output is less noisy and detected objects are more filled.

4.2.2 "Winter" Sequence

In the subjective performance evaluation of this sequence we note the following:

i) The same trend (see "i" Sec. 4.2.1) of bootstrapping recovery is repeated as for the "Fog" sequence, however, due to the slower moving objects of this sequence, a more clear ghost is present in the output of [8] (Fig. 4.5).

ii) The more correct segmentation output was obtained by the proposed technique.

iii) The output of the proposed technique was better than that of the reference techniques. However, due to sever cracks present in detected foreground blobs in all outputs (Figs. 4.5,4.6), each object appears as multiple objects, which may lead to failure in following higher video processing stages.

4.2.3 "Snowing" Sequence

In the subjective and computational performance evaluation of this sequence, we note the following:

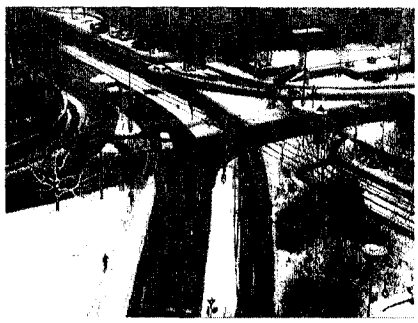
i) The superiority of the proposed techniques was evident in all the frames as can be seen in the sample results of Figs. 4.7, 4.8.

ii) In many frames the sever cracks in detected foreground blobs lead to multiple false object representation at the final output of both the reference and proposed techniques (Figs. 4.7, 4.8), despite the better performance of the proposed segmentation technique compared to the other reference techniques.

4.2.4 "Rene-Levesque Guy" Sequence

With this sequence we evaluated the algorithms performance in addressing problems that were not embedded in any of the previous sequences, mainly the waving trees (temporal clutter) with large pixel displacement and the temporary stopped foreground objects (sleeping person).

i) The multi-component (background update) nature of the proposed technique and that of [8], significantly reduced the effect of multi state background pixels corresponding to perturbing tree leave especially near tree boundaries, given that enough



(a) Original "Winter", I_{119}.

(b) Output of [23].

(c) Output of [24].

(d) Output of [8].

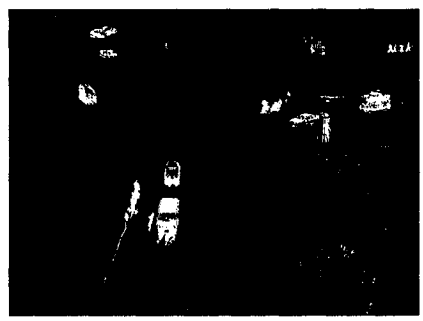
(e) Output of proposed technique.

Figure 4.5: Algorithm outputs for sample frame I_{119} of "Winter" sequence. for (d) the output contains false detected ghost objects due to slow bootstrapping recovery, these ghost overlap with the detected originals object thus appearing more filled compare to other reference techniques and the proposed.

(a) Original "Winter",I_{249}.

(b) Output of [23].

(c) Output of [24].



(d) Output of [8].

(e) Output of proposed technique.

Figure 4.6: Algorithm outputs for sample frame I_{249} of "Winter" sequence. Proposed technique output out-performs the reference techniques.

(a) Original "Snowing", I_{107}.

(b) Output of [23].

(c) Output of [24].

(d) Output of [8].

(e) Output of proposed technique.

Figure 4.7: Algorithm outputs for sample frame I_{107} of "Snowing" sequence. Proposed technique output achieved better object detection with clearly less noise.

(a) Original "Snowing", I_{214}.

(b) Output of [23].

(c) Output of [24].

(d) Output of [8].

(e) Output of proposed technique.

Figure 4.8: Algorithm outputs for sample frame I_{214} of "Snowing" sequence. Proposed technique output more correctly detected object with less noise and no false ghost objects. In (b) a ghost corresponding to the tram falsely detected (sleeping person). For (d) more ghost objects are falsely detected due to both sleeping person and bootstrapping problems.

sample frames of each state were gathered. The other two single-model based techniques of [23, 24], produced in many frames noise like false detections (Fig. 4.9).

ii) The periodical component weight normalization in the proposed technique, prevented traffic-light stopped vehicles from getting merged into the background model even with limited initial statistics (Fig. 4.9). Other reference techniques not only merged these vehicles into their background models, but created temporary false ghosts in the foreground mask when these object start to move again (green light).

iii) Also for this sequence, the proposed technique more accurately detected moving objects compared to the other reference techniques.

4.2.5 "St-Catherine" Sequence

With this sequence we evaluated algorithms adaptation rate, object detection accuracy and the performance in addressing the temporal clutter problem (waving flag).

i) The multi-component (background update) nature of the proposed technique and that of [8], significantly reduced the effect of multi state background pixels corresponding to waving flag. The other two single-model based techniques of [23, 24], produced in many frames noise like false detections (Fig. 4.10).

ii) The proposed technique had better object detection accuracy with more filled objects, less shadows and noise compared to the other reference techniques.

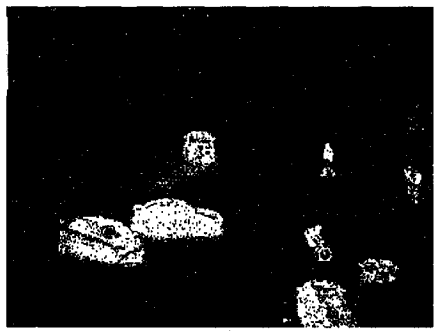
4.3 Computational Performance

We present the average computational time required by each algorithm when simulated on the first 100 frames of each test sequence. Note that for this evaluation only, we used an update interval $\Delta t = 3$ and a median buffer length $L = 30$ for [23], in order to simulate with a full buffer. For [14,24], we kept the update interval $\Delta t = 10$, since it is very computationally demanding. For the proposed technique we used the hysteresis thresholds $\lambda_1 = 3.0$, $\lambda_2 = 2.0$.

As shown in (Fig. 4.11) both the proposed technique and that of [23] in average have similar computational performance. Note that the object detection accuracy of

(a) Original Rene-Levesque and St Guy crossing, I_{819}.

(b) Output of [23].



(c) Output of [24].

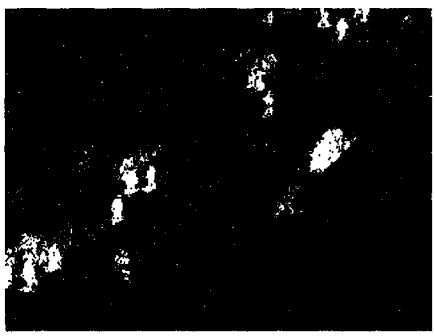
(d) Output of [8].

(e) Output of proposed technique.

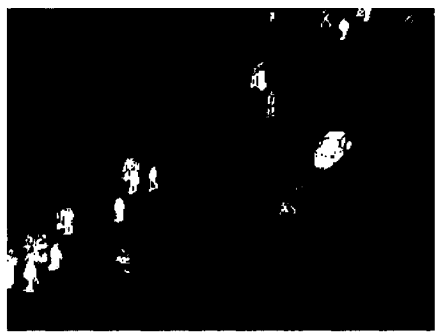
Figure 4.9: Algorithm outputs for the "Rene-Levesque Guy" crossing sequence. In (b), (c), (d) two temporary stopped vehicles are falsely merged into the background. For (b) false detection of shallow waving tree leaves (middle-right). For (b), (d) a ghost corresponding to moved vehicle (at the top) is falsely detected. For (c) more false noise is present and detected objects are less filled. For (e) the proposed technique output has less noise with no false ghosts and no vehicles that are wrongly merged into the background model.

(a) Original St-Catherine, I_{907}.

(b) Output of [23].



(c) Output of [24].



(d) Output of [8].

(e) Output of proposed technique.

Figure 4.10: Algorithm outputs for the "St-Catherine" sequence. In (b), (c), (d) varying degree of false detections related to the waving flag (left-bottom of the satellite receiver in center of the frame). For (c) more false noise is present and detected objects are less filled. For (e) the proposed technique output has less noise, more accurate object detection and negligible waving flag related false detection.

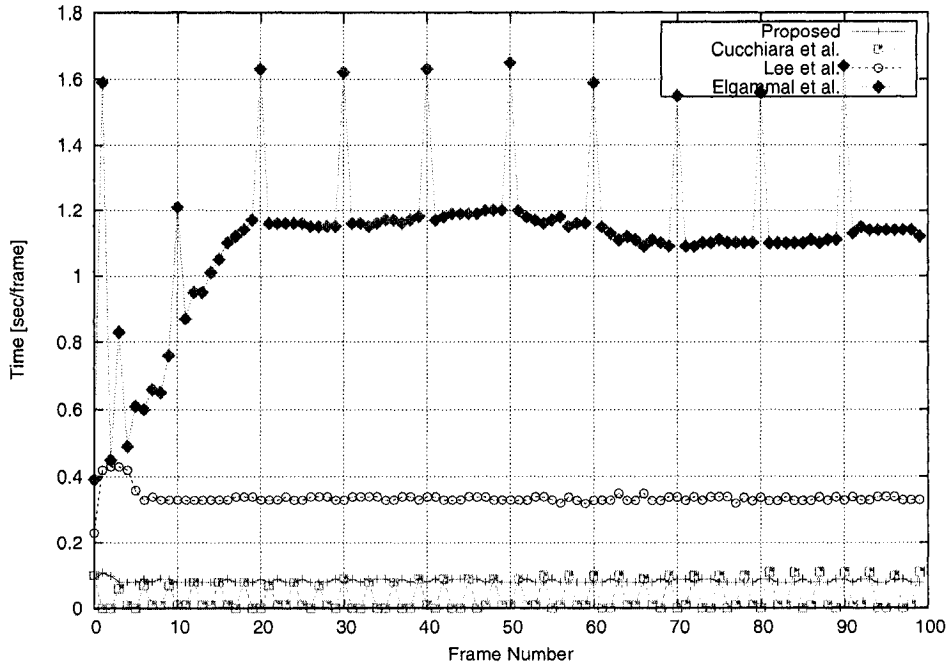


Figure 4.11: Average computational performance chart for the first 100 over all the test sequences.

the proposed technique outperform that of [23] (Figs. 4.1,4.2). The MOG-framework based technique of [8] in average was close to real time. The KDE-based technique of [14, 24] is critically slow and computationally demanding.

4.4 Practical Considerations

The proposed techniques consists of various schemes (modules), each with its corresponding parameters that effect the overall performance. In system implementation (Sec. 3.5) we used a set of parameter values that generally performed very well in many sequences, however depending on the current application (monitored scene), even more better performance can be obtained by careful considerations of the following:

i) The choice of hysteresis thresholds λ_1, λ_2 : For scenes with small foreground moving objects, typically when objects and their cast shadows has size less than 10% of frame size (Fig. 4.12), it is good to use $\lambda_2 \in [1.0 - 1.5]$ and $\lambda_1 \in [1.5 - 2.0]$. Note that λ_2 corresponds to the base threshold that controls the minimum intensity

Figure 4.12: Foreground objects of 10% frame size.

Figure 4.13: Foreground objects larger than 20% frame size.

difference required for being a background pixel. While λ_1 generally controls the amount of small cracks and added shadows in/to foreground blobs.

In the case of large foreground objects, size larger than 20% of frame size (Fig. 4.13), and if motion history is used it is good to use $\lambda_2 \in [2.5 - 3.0]$ and $\lambda_1 \in [4.5 - 5.0]$. If motion history is not used then $\lambda_2 \in [2.5 - 3.0]$ and $\lambda_1 \in [3.0 - 3.5]$.

ii) The state of motion history scheme: Generally the motion history always helped in detecting moving objects boundaries, however in the case of very fast moving objects, where they typically stay in the scene for two to three frames only (Fig. 4.14), motion history creates lagging tails connecting close foreground objects together that form a continuous false foreground region. The other case were motion history should not be used is when object detection with precise boundary is required.

iii) Weight normalization threshold / ratio: For any pixel, given that the real background was obtained, these two parameters determine the time needed for a stationary foreground object to get merged and become the new background. More importantly, proper choice of these parameters will prevent unwanted merger of tem-

(a) Autoroute Ville-Marie, frame 1254.

(b) Autoroute Ville-Marie, frame 1255.

Figure 4.14: Very fast moving truck.

porary stopped foreground objects, i.e. if some foreground objects frequently become static for 50 *seconds* corresponding to $T_{stop} = 50 \times 30\, fps = 1500\, frames$, then the normalization threshold must be $w_{norm} > \frac{2T_{stop}}{(\lambda_{bf}+0.5)}$ where 0.5 is error margin corresponding to foreground object stop time tolerance.

Chapter 5

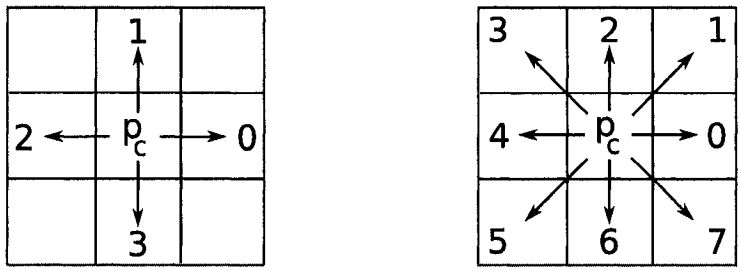
Contour Tracing and Filling

In this chapter, we introduce methods for grouping and identifying the connected regions in the initial foreground binary masks. Component-labeling algorithms are the most basic and commonly used techniques that consists of either region growing or contour tracing and filling.

Region growing techniques [55–59] need an interior starting point (seed pixel) and a recursion-based growing procedure, this category of techniques are more robust but computationally demanding and require more storage compared to the other techniques. Contour tracing and filling based techniques are more widely used techniques, since they adhere to the real-time and limited storage criteria necessary for practical on-line video processing applications.

5.1 Introduction

Contour tracing and filling are fundamental processes in various image and video applications. Most of the presented tracing and filling algorithms are pixel or raster-based, while there are few which are polygon or vector-based, also known as "Ordered Edge List" [60,61]. In raster-based, the image or video frame is stored as a matrix corresponding to pixel luminance or color values. In vector-based, raster image or video frame information is represented as geometrical data corresponding to the shapes in the image plane, such as lines, curves or simple shapes. Then from these geometrical



(a) A 4-neighborhood. (b) A 8-neighborhood.

Figure 5.1: The neighborhood of a pixel p_c.

information raster representations can be produced for visualization.

In an image defined on a rectangular sampling lattice, two types of neighborhoods are distinguished (Fig. 5.1): 8-neighborhood and 4-neighborhood. In an 8-neighborhood all the eight neighboring points around a point are considered. In a 4-neighborhood only the four neighboring points, right, left, up, and down, are considered.

A contour is a finite set of points, $p_1, \ldots, p_n$, where for every pair of points p_i and p_j in C there exists a sequence of points $s = \{p_i, \ldots, p_j\}$ such that i) s is contained in C and ii) every pair of successive points in s are neighbors of each other. Any contour can be represented by the point coordinates or by a chain code.

Contour tracing is an algorithm that groups neighborhood connected pixels in a binary edge image $E(n)$. Contour filling fills the inner pixels of a contour and the pixels of any gaps in that contour with specific gray-scale values.

Schemes for contour tracing in raster graphics are broadly divided into two categories: edge-based and binary-based. Edge-based schemes rely on some sort of edge detection schema, whether it is binary-based as in [62], or the more general second-order zero-crossing of Laplacian or Canny [31]. Edge pixels are then traversed and tracked producing fully connected sequences of pixels that define the outer object (or inner gap) contours. Binary-based schemes [27, 28, 30] work directly on the binary or segmented image or video frames, by tracing the outer or inner boundary of each segmented blob. In both approaches, many algorithms try, while tracing, to label traversed outer or inner boundary pixels with different labels (e.g. traversal chain

code), to be used later for either efficient storing or for contour filling.

Contour filling can be classified into two categories: label-based filling, and parity-check filling. In label-based filling [27–30] the pixel label information (e.g. chain-code) obtained during tracing is used for filling. Parity-check filling [26] techniques are generally faster and require less memory, however, they may leak in the case of complex object contours.

5.2 A Survey of Related Work

In [25, 26], Pavlidis presented algorithms for both contour tracing and filling. The contour tracing is based on the 8-neighbor connectivity of a pixel in an edge-detected image and the contra-clockwise traversing concept for locating the next connected neighboring pixels. The traversing consists of two phases: external or region contour traversal and internal or inner contour traversal. Graph traversal, is used to extract the LAGs (line adjacency graphs) as first presented by [63]. Pavlidis extended the LAG definition into c-LAGs for contour pixels, and i-LAGs for interior (background) pixels. After extracting the c-LAGs during the contour tracing phase, the filling algorithm uses the notion of parity-check or the LAG degrees to locate candidate horizontal seeds, then it uses the notion of connectivity or seed growing to fill the gap until the other horizontal point of the c-LAG pair is reached. In the case of reaching an internal c-LAG during the filling, the algorithm starts filling the gap with different gray scale. The tracing algorithm presented by Pavlidis, is an effective, fast and one of the few algorithms that handle inner contours that share some of its points with the bounding outer contour. However, both the tracing and filling algorithms fail in most contours that have complex concavities and multiple cross points.

The tracing and re-filling (or re-labeling) algorithm of Chang et al. [27, 28] starts by first locating a non-labeled foreground pixel in a binary image or frame, upon which the Tracer procedure is called, to start the tracing clockwise and labeling of the outer boundary contour pixels of the region. After tracing the outer contour the filling procedure starts labeling the interior pixels of the extracted outer contour with

the same label as that of the outer contour pixels. During the filling operation, if an unlabeled background pixel is detected, the Tracer procedure is called to trace in a contra clockwise manner and label the interior contour pixels (the contour of a gap in the current region). After closing the interior contour the filling procedure continues again. The Chang et al. algorithms, do not need an edge detection of the binary or segmented images or video frames and require only one image pass for both tracing and filling. However, the algorithm is highly storage and computational demanding and not suitable for real time video applications.

The work of Codrea et al. [29] is a detailed algorithm for both contour tracing and filling that works on binary edge images. The contour tracing of Codrea et al. [29], starts by scanning the image from left to right, top to bottom. After finding the first contour pixel (starting point of the contour), next contour pixel is located by contra-clockwise scanning of the 8-neighborhood of the current pixel starting at the pixel positioned in the direction $(dir+7)\ mod\ 8$ for even direction values and $(dir+6)\ mod\ 8$ for odd direction values, until the first two points with the same sequence order of the contour reached again. The direction variable is initialized to $dir = 7$, and updated before each search for the next contour point, depending on the locations of both the current and previous contour points. While tracing, each contour point is labeled with one of four different labels R, L, T, B, and stored in an accompanying separate matrix. The labeling is done in one image pass via two different procedures : single direction-based labeling and double directional-based labeling. After finishing tracing and labeling, contour filling scheme is straight forward: it scans the labeled region contour pixels horizontally line by line, filling between pairs of $L - R$ pixels only. The algorithm presented by Codrea et al. is fast and effective, however, it has filling problems in inner contour gaps and not suitable for video processing.

The proposed contour tracing scheme that is based on [2] takes as an input a binary edge image. But, unlike related algorithms, it addresses the following important key issues at each new candidate contour pixel: i) detection and deletion of contour dead branches, ii) detection and exclusion of unclosed contours that resulted due to faulty segmentation, noise, occlusion, iii) detection of inner hole contours, and vi) efficient

extraction of contours and their specific features such as compactness while tracing the edge image, and later storing these features with the extracted contour points as independent objects for subsequent processing stages.

The proposed contour filling scheme combines the simplicity of a label-based and the robustness of seed growing techniques. Labels are obtained during the tracing phase, corresponding to the chain-code or directions of each contour point. Then these points in a given scan-line of the contour are used as either a terminating or line growing seeds depending on their label information, insuring that the algorithm can handle all subtle cases of complex concavities, multi-crossing edge points and parallel edges.

5.3 Improved Contour Tracing Algorithm

In this section, we propose improvements to the tracing scheme in [2]. These improvements are; i) new chain-code based candidate-point search mechanism (Sec. 5.3.3) for improved performance, ii) new added functionality (Sec. 5.3.5) to address cross-point connected closed contours (Fig. 5.6).

The proposed algorithms require as an input a binary edge image E_t for each video frame at time instance t. This binary edge image can be obtained in various ways. We used the method in [51,62] to extract edge information. Fig. 5.2 shows the block diagram of the proposed set of algorithms.

Following the edge extraction, the contour tracing algorithm is called for each edge frame E_t at time instance t. The result of tracing edges is a list C_t of contours and their features: starting point, perimeter, width, height, area, center of gravity, compactness, extent ratio, and irregularity ratio. Tracing is done in a contra-clockwise manner and for each current point p_c in E_t the algorithm looks for a neighboring point in the 8-neighborhood, always starting at the rightmost neighbor. This rule forces the tracing algorithm to move around the object by looking for the rightmost point and never inside the object. At each new contour point, the algorithm records both its co-ordinates and tracing directions (chain-code).

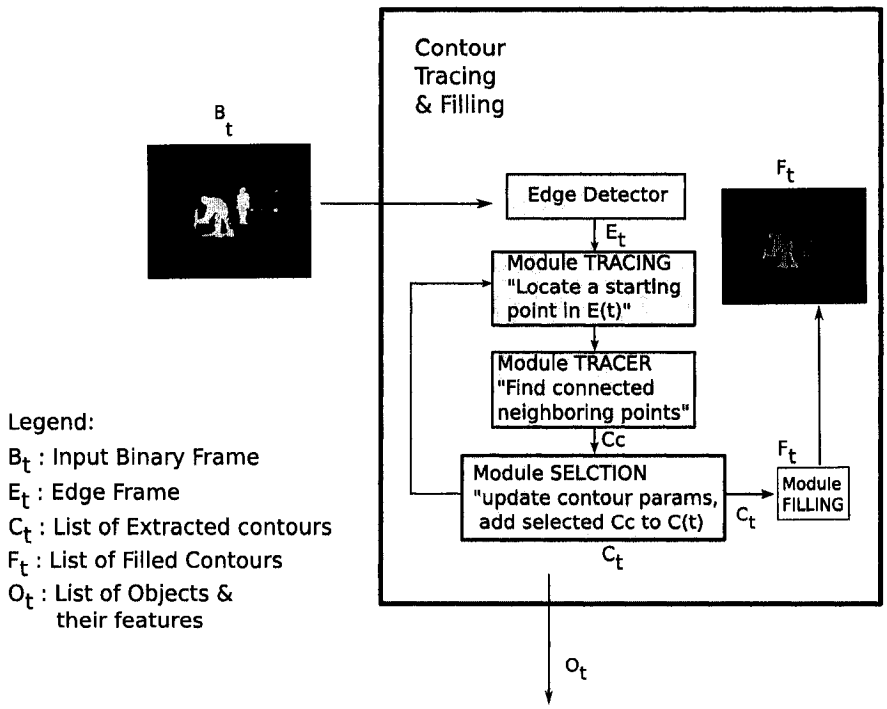


Figure 5.2: Proposed tracing algorithm block diagram.

5.3.1 Notation

In the following, let

- C_t be the list of contours in the original video frame I_t at time t,
- $C_c \in C_t$ is the current contour with starting pixel p_s,
- MBB_c is the minimum bounding box of C_c,
- p_c is the current processed pixel,
- p_i is an 8-neighbor of p_c, and p_n corresponds to the next candidate contour pixel of p_c,
- d_s is the search direction for the next contour pixel candidate, $p_i(d_s)$ is the 8-neighbor of p_c at location corresponding to d_s (Fig. 5.1-b),
- contour pixels p_s, p_c, p_i and p_n are structures of three properties; x co-ordinate, y co-ordinate and the chain-code info cc_s.

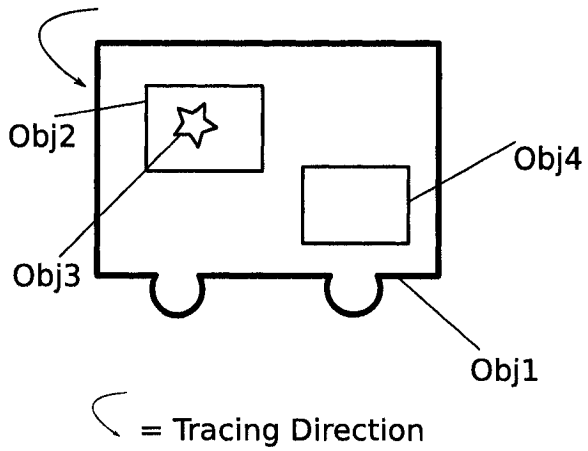


Figure 5.3: Scanning Direction.

Only co-ordinate information (x, y) is considered when comparing two contour pixels $p_n = p_i$ or $p_c \neq p_s$, while all properties of co-ordinate and chain-code information are set when matching $p_c := p_s$. In E_t, background pixels p_b have the value of zero (black), and foreground edge pixels p_w can have any given gray-scale value other than zero.

The proposed tracing scheme consists of five integrated modules: i) tracing module, ii) tracer module, iii) dead branch deleting module, iv) connected closed contour module, v) internal contour check module. The functional details of each module are presented next.

5.3.2 Module Tracing

Scan E_t from top to bottom and left to right until finding a *non-visited* edge pixel p_w. Upon finding p_w, set $p_s := p_w$, set $p_c := p_s$ and apply *Module Tracer*. If no more p_w pixels can be found then end the tracing in E_t. Note that, in the case of objects contain other objects, the given scanning direction (see Fig. 5.3) forces the algorithm to trace first the outwards and then the inwards contours.

5.3.3 Module Tracer

Track edge pixels contra-clockwise in order to find the neighboring points of p_c as following

1. Set the search direction $d_s := 5$, which corresponds to the bottom-left neighboring pixel of p_c

2. Verify that p_c is not an isolated edge pixel, by traversing all the surrounding pixels:

   - if an edge pixel $p_i(d_s)$ is found: i) update the chain-code of p_c to $cc_s := d_s$ and add p_c to C_c, ii) set $p_c := p_i$, iii) update d_s as following: if $d_s = 0\,or\,1$ then set $d_s := 7$; if $d_s = 2\,or\,3$ then set $d_s := 1$; if $d_s = 4\,or\,5$ then set $d_s := 3$; if $d_s = 6\,or\,7$ then set $d_s := 5$, iii) execute step (c).
   - if no neighboring edge pixel of p_c is found, then p_c is an isolated edge pixel, set this pixel to zero (background) in E_t, return to *Module Tracing* to find next *non-visited* p_w.

3. While $p_c \neq p_s$ repeat the steps in the block diagram shown in Fig. 5.4.

4. When $p_c = p_s$ then C_c is closed, update the MBB_c, contour size, width, height, area, extent ratio, compactness and irregularity ratio of C_c.

5. Perform *Module Check Internal* that checks if C_c is internal.

6. Add C_c to C_t, if its properties satisfy some given criteria:

   - C_c has a width or a height larger than specific thresholds T_w and T_h, or
   - C_c has a perimeter larger than a specific threshold T_p , or
   - C_c satisfies a given irregularity ratio T_i.

7. Return to *Module Tracing* to find next *non-visited* edge pixel p_w.

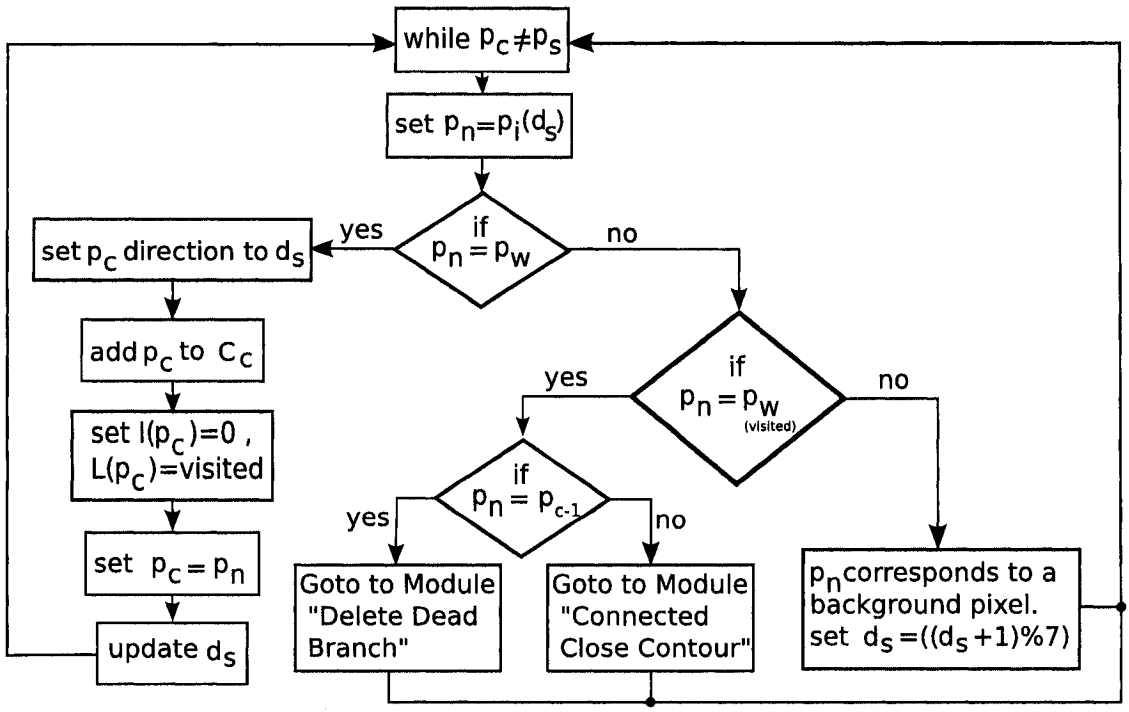


Figure 5.4: Module-Tracer block diagram.

It is very important to update the chain-code information cc_s of each p_c prior to adding them to C_c. This chain-code information should always correspond to the chain-code relative to previous added p_c.

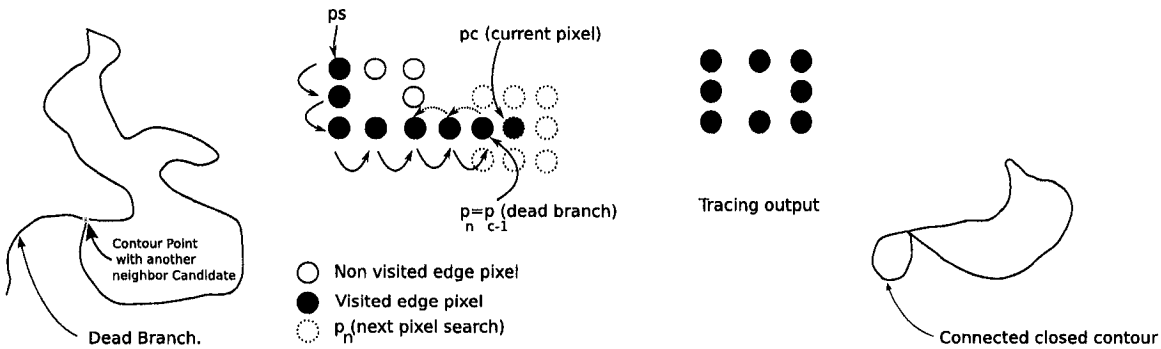
5.3.4 Module Delete Dead Branch

Delete dead branches (Fig.5.5-a,b) and return to *Module Tracer.*

1. Let p_l correspond to the last contour point in C_c.
2. Keep removing p_l from C_c until finding another neighboring candidate edge pixel p_w in E_t.
3. Upon finding p_w set $p_n := p_l$ and return back to *Module Tracer.*

5.3.5 Module Connected Closed Contour

Remove any connected closed contours to C_c (Fig. 5.5-c) and then add them to C_t.



(a) Contour with dead branch.

(b) Connected closed contour.

(c) Cross-point connected closed contour.

Figure 5.5: Module Trace sub-cases illustrated.

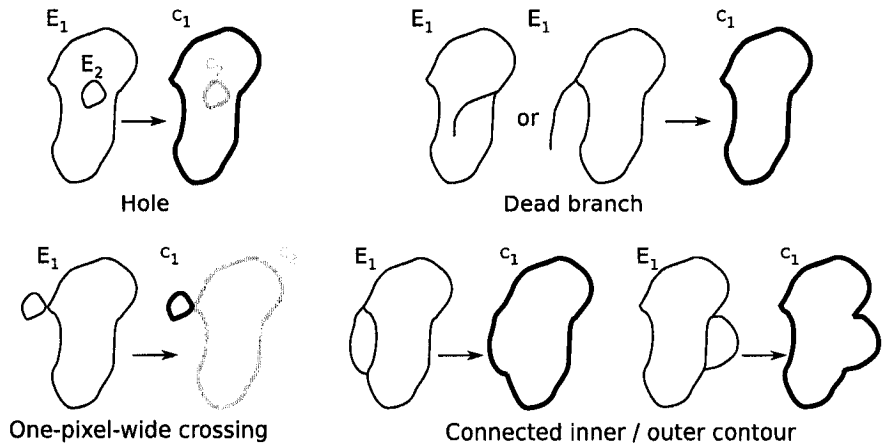


Figure 5.6: Special case contours illustrated.

1. Let C_{tmp} be a temporary contour

   (a) Start from p_c and add all the contour points in C_c to C_{tmp} until reaching the contour point in C_c corresponding to p_n.

   (b) Update the MBB_{tmp} of C_{tmp}.

   (c) Perform *Module Check Internal* that checks if $C_t$is internal.

   (d) Update C_c properties and add C_{tmp} to C_t and return back to *Module Tracer*.

Note that the given scanning direction and the addition of connection pixels (one-pixel-wide case) to both C_{tmp} and C_c in the *Module Connected Closed Contour*, produces the trace-outs contours shown in Fig. 5.6.

5.3.6 Module Check Internal

Verify if the starting point p_s of a given contour C_c inside any of the contours $C_i \in C_t$. Let f_i be a flag and for each C_i:

a) Set $f_i = false$.

b) If p_s of C_c is not inside any MBB_i of C_i then return and designate C_c as non-internal.

c) If, however, p_s is inside a MBB_i:

- Create an array corresponding to a single scan-line in MBB_i of C_i, set all the values in this array to eight (or any other value larger than the largest chain-code seven).

- Locate those contour point in C_i with the same y co-ordinate of p_s, and set the corresponding x co-ordinate locations in the array to those contour points with directional information d_s (see Fig. 5.7), however, if the directional information d_s of these contour points: i) zero or one: then check the d_s of the following contour point in C_i, if it is seven then set the corresponding x co-ordinate locations in the array to seven, if not seven set the location in the array to the actual directional d_s value, ii) six or seven: then check the d_s of the following contour point in C_i, if it is one or two then set the corresponding x co-ordinate locations in the array to one, if not one set the location in the array to the actual directional d_s value.

- Scan each array location, if a value less than eight is found : i) set $f_i = true$ if this value is five, six or seven, if not set $f_i = false$, ii) change the value in the current array location to ten if f_i was true, iii) check the array location corresponding to the x co-ordinate of the p_s of C_c after scanning all the array, if it is equal to ten then C_c is internal to $C_i \in C(n)$.

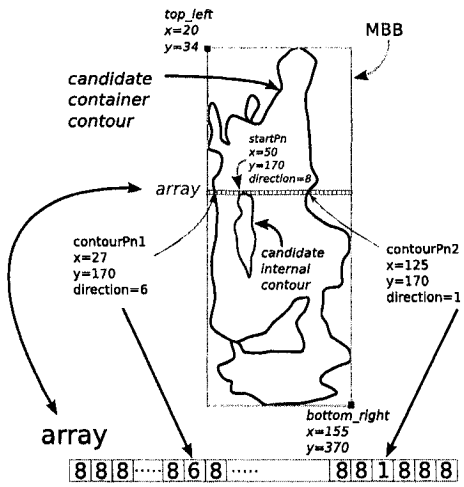


Figure 5.7: Internal checking mechanism in Module Check Internal.

5.4 Proposed New Filling Algorithm

After selective tracing of the valid closed contours C_t, the proposed novel filling algorithm fills every contour $C_i \in C_t$, based on contour point's chain code information obtained during tracing. Let F_t be the list of filled contours, $F_i \in F_t$ correspond to a filled contour, $ccp_i \in C_i$ is the current contour point, $ccp_{i+1} \in C_i$ is the next contour point. Fill C_i corresponding to F_i only for the following two cases for each ccp_i:

1. If the chain-code cc_i of ccp_i is $\{5$ or 6 or $7\}$ **and** cc_{i+1} for ccp_{i+1} satisfies $(cc_{i+1} > cc_i \, mod \, 5)$, then use the point to the right of ccp_i as a seed, filling rightwards until reaching the next contour point in the same scan-line.

2. If the chain-code cc_i of ccp_i is $\{0$ or $1\}$ and $cc_{i+1} = 7$, then use the point to the right of ccp_i as a seed, filling rightwards until reaching the next contour point in the same scan-line.

Fig. 5.8 illustrates the only four different cases where filling should take place depending on the chain-code information cc_i of each contour point. For example, a contour point with $cc_i = 6$ (second row in Fig. 5.8) filling rightwards should start only if $cc_{i+1} = 5$ or 6 or 7. Other cases of cc_{i+1} correspond either to special cases of sudden change in tracing direction $cc_{i+1} = 0 \, or \, 1$ (last two cases of the second row in Fig. 5.8), or to non-existent contour points due to the nature of contra-clockwise tracing

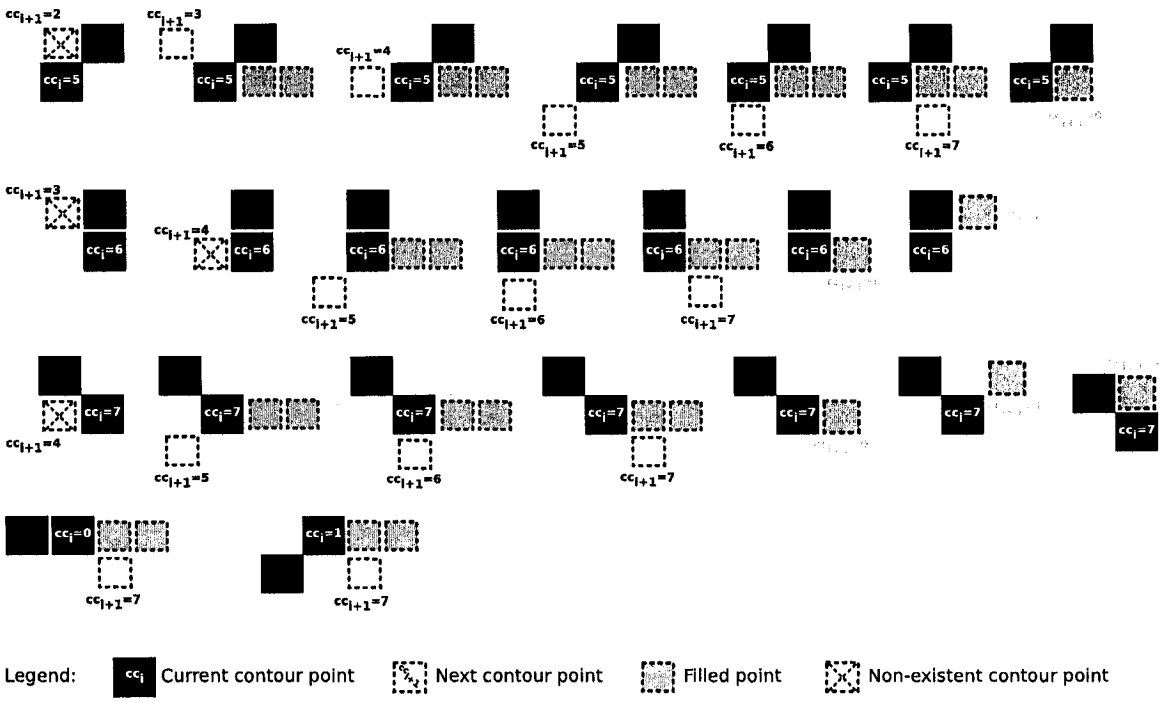


Figure 5.8: Filling cases illustrated.

mechanism (first two cases of the second row in Fig. 5.8). Our implementation of the proposed simple filling algorithm is efficient and is on average three times faster than the reference [25, 29] filling algorithms.

5.5 Analysis of Proposed Contour Tracing and Filling

In this section, we analyze and compare the accuracy, performance and efficiency of the proposed tracing and filling algorithms with Pavlidis [25, 26], considered as reference set of algorithms, and Codrea and Nevalainen [29], a recent promising scheme of algorithms.

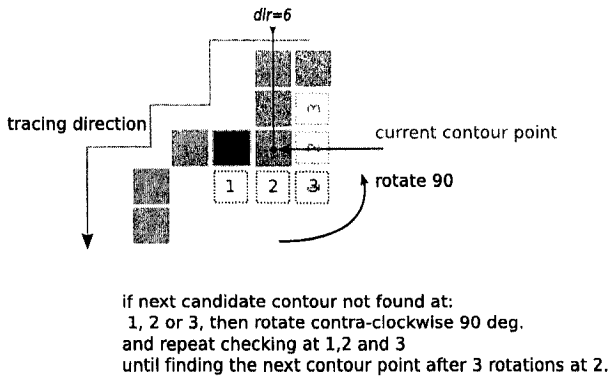


Figure 5.9: Un-efficient next contour points search mechanism by Pavlidis.

5.5.1 Functional analysis

We analyze the functional concept of the two reference algorithms and investigate their convenience for real time video applications.

The work of Pavlidis

The Pavlidis tracing algorithm has the following main performance functional drawbacks:

- The first one is related to the search technique presented for finding the next contour point in the edge image, while searching for the next candidate contour point, if it is not found at the current search row $1, 2,$ *or* 3 in Fig. 5.9, then the search row is rotated 90° left (or right in case an inner contour is traced), and the surrounding locations of the current pixel is searched again at the new rotated $1, 2,$ *or* 3, this leads to checking the same location twice. This location corresponds to 3 in the previous row before rotation, and corresponding to 1 after rotation, that is; first it checks locations 1, 2, 3 then $3_{before\,rotation} \longrightarrow 1_{rotated}$, $2_{rotated}$, $3_{rotated}$, so each time there is rotation in the search procedure, the last location corresponding to 3 is checked twice.

- Another important issue related to the search mechanism, is that depending on the current search direction d_s, the search for the next contour point always starts with either the leftmost (or rightmost) pixel in the row perpendicular

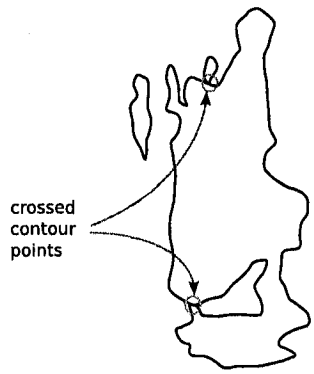


Figure 5.10: Crossed contour points of a real video object.

to the current search direction d_s, and in many practical contour cases the next point is usually located just above the first search location in the row (see Fig. 5.9). This means that the algorithm has to look into eleven locations (3 rotations) before finding the next contour point.

- No mechanism for checking whether a contour is internal or not, although Pavlidis presented in [26] a way for detecting specific internal contours that share some of its points with the outer contour, although it was not clearly stated and it caused filling errors.

- In various cases of contours that have crossed points (see Fig. 5.10), the algorithm fails to correctly extract the single closed contour, instead it separates the contour into more than one connected contours which leads to leaks in the filling process.

- Fundamentally not suitable for object-based video applications. Since there is no mechanism for detecting connected dead branches or connected closed contours, thus producing contours that are not representative of the actual real video objects. This non-representativeness corresponds to contours that have very different features of video object shape, width, height, area, compactness and irregularity ratios.

The presented *parity check* scan-line based Pavlidis filling is fast but fails with complex contours, especially in the presence of cross contour points. Also, the lack

of internal contour check mechanism, lead to the treatment of inner hole contours as other external contours which not only wrongly filled these contours but also produced unwanted leaks in the outer containing contours.

The work of Codrea and Nevalainen

The tracing algorithm of Codrea et al. has an efficient technique to search for the next contour points. It also has a special contour search termination condition: *"finding the first two contour points again with the same sequence of occurrence as the one obtained when first started the tracing" [29].* The algorithm succeeded in blindly tracing the whole contour regardless of the contour's complexity, but has the following main drawbacks:

- It may trace the same set of contour points more than once before reaching the termination condition. This often occurs when the video scene contains many foreground objects with complex contours that have dead branches or in the presence of many unclosed contours due to noise. This greatly increases the tracing time and storage necessary for storing the extracted contours.

- There is no mechanism for checking if the contour extracted is located inside another already detected contour. This is important in various cases, if the contours are of large size then they correspond to inner object gaps and should not be treated as different independent contours. If these inner contours are of a small size then mostly related to noise or segmentation errors and should be rejected and removed.

- The absence of a mechanism to detect the presence of connected dead or inner branches and connected closed contours makes the tracing algorithm fundamentally not suitable for object-based video applications.

The *Label*-based filling algorithm by Codrea et al. is simple and robust. It is rather done in the same way of scan-line filling, but instead of *parity-checking* pair calculation as it was the case in Pavlidis's filling schemes, it relied on the special labels extracted

during the tracing. Also, here the lack of internal contour check mechanism lead to wrongly filled inner gaps and leaks in the outer containing contours.

The proposed algorithms

The advantages of our tracing for object-based video applications can be summarized with:

- It has an efficient conditional-based mechanism (see *Module Tracer*) for updating the search direction d_s used to locate the next contour points. This insures that the tracing algorithm always moves around the objects and never inside the objects.

- It never re-traces the same contour points twice, and deletes while tracing any connected with noise or occluded objects.

- It effectively addresses some of the most important issues for object-based video applications. These issue are connected closed contours, dead or inner branches and inner gap contours. Connected closed contours are detected and extracted correctly as shown in Figs.5.10, 5.6. Dead or inner branches are detected and removed, which not only produces more corresponding contours, but also greatly reduces tracing time especially in complex video sequences. Valid large-enough inner gaps contours are correctly related to the containing contours, while discarding those small noise or faulty segmentation related inner contours.

- Every extracted contour is stored and processed as an independent object with the features: starting point, minimum bounding box, width, hight, size, center of gravity, internal state, area, compactness, extent ratio and irregularity ratio. These features are calculated and updated while tracing. This is important step since these features are used in various subsequent processing stages such as video object tracking, video event analysis or video object classification.

- The proposed filling algorithm is contour-based since only the interiors of each contour is scanned instead of redundantly scanning the whole frame as it is in

the reference filling algorithms [25,29]. The contour-based filling not only makes the proposed filling simple and effective, but also ensures that it never leaks.

5.5.2 Complexity and efficiency

We analyze the complexity of the proposed tracing and filling algorithm through the following features:

- *Proposed tracing algorithm visits each pixel in the input edge frame E_t only once.* Since each input edge frame is scanned once, and all the encountered edge pixels are labeled to *visited* , regardless if they belong to a connected dead or inner branches or other connected contours, this insures that each edge pixel is visited only once. Unlike the other related tracing algorithms each edge pixel may be visited more than once especially in complex scenes. And since the search for each next contour pixel requires a variable amount of time, this makes our tracing algorithm more efficient.

- *Proposed tracing algorithm is linear in time.* The proposed tracing algorithm proceeds in a contra-clockwise manner, moving linearly in time pixel by pixel starting from the first non visited edge pixel, until re-reaching the same starting point or not finding anymore non-visited edge pixels. This means that the tracing time for a given set of contours is always the same regardless of their locational distribution.

- *Proposed filling algorithm traverses only the inner part of each extracted contour (contour-based), labeling each traversed inner pixel only once.* Since the filling operation for each contour is accomplished by scanning only the projection of each contour from top to bottom and left to right, labeling each traversed inner pixel only once according to the label information of the contour points along that filling line, thus *visiting* and labeling each inner contour pixel only once. On the other hand, all the other presented filling algorithms including the two reference algorithms require scanning the whole contour image for filling and

frame size	Pavlidis	Codrea et al.	Proposed
352 × 288	0.0476 s/f	0.0488 s/f	0.0475 s/f
320 × 240	0.0378 s/f	0.0382 s/f	0.0344 s/f

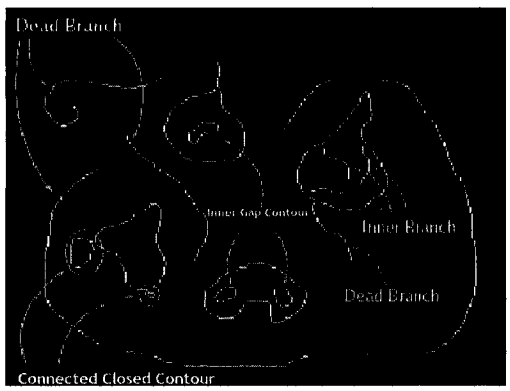
Table 5.1: Average tracing and filling times in seconds per frame.

not only the projection of each contour, which adds extra overhaul to the overall filling operation.

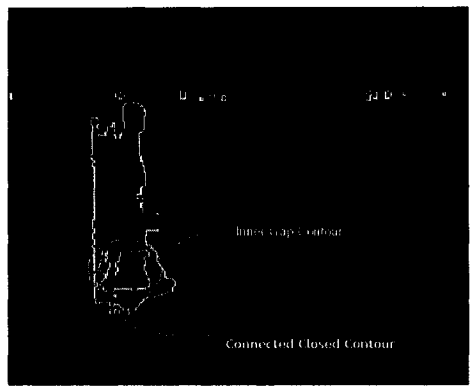
The algorithms were implemented in C++ on a single processor Linux based PC with Intel P4@2.4GHz and 768MB of memory. All reference and presented tracing and filling algorithms run in real-time with an average time that depends on the video scene complexity (Table 5.1). As the scene complexity increases and more other processing stages are added to the system, the efficiency of the proposed tracing scheme becomes more clear, as the number of false extracted video objects is greatly reduced. For example, given that a non closed contour is extracted and added to true video contour list by the two reference algorithms, this will lead to great computation overhead in the following tracking stage while trying to match this video object or make a correspondence with the other valid video objects in this frame. Note that the proposed filling algorithm is on average three times faster than the reference methods. The speed ratio of the proposed filling to tracing is 1:32.

5.5.3 System Implementation and Parameters

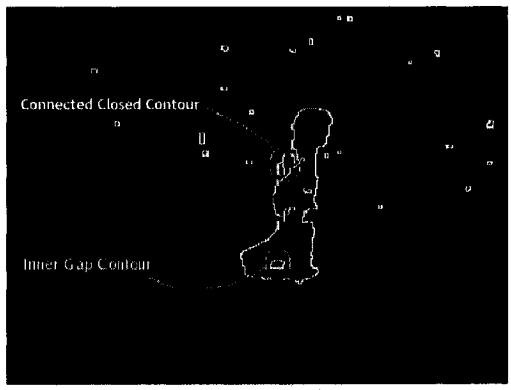
The proposed contour tracing and filling scheme implementation is straight forward since it is based on binary frames. We used a value of 250 for foreground binary pixel representation (white blobs), background pixels were represented by zero intensity values. In Module Tracer, we marked visited edge pixels with intensity value of 251 (any other value can be used, except for zero and 250). The only non-automated parameter is the minimum contour size, which is chosen based on the desired size of objects to be included in the final list of contour objects.



(a) E(n) of "Synthetic Image".



(b) E(n) of "Hall".



(c) E(n) of "Meetingroom".

Figure 5.11: Complex contours indicated on the "Synthetic Image" and sample frames of "Hall" and "Meetingroom" sequences.

5.6 Experimental Results

We evaluate the performance of the tracing and filling algorithms based on their contour representativeness for object-based video processing. We present objective and subjective results obtained by applying the proposed scheme of algorithms and those of Pavlidis's and Codrea *et al.*'s on four widely used video sequences, indoor: "Hall" and "Meetingroom", outdoor: "Survey" and "Highway". These sequences contain scenes with varying complexity, depending on the moving foreground objects and the illumination conditions. We also test the algorithms on a synthetic image (Fig. 5.11-a) created to show tracing and filling problems related to complex contours in real-world sequences. Some of these problems are connected closed contours, dead, or inner branches, and inner hole contours as can be seen in Fig. 5.11.

5.6.1 Objective evaluation

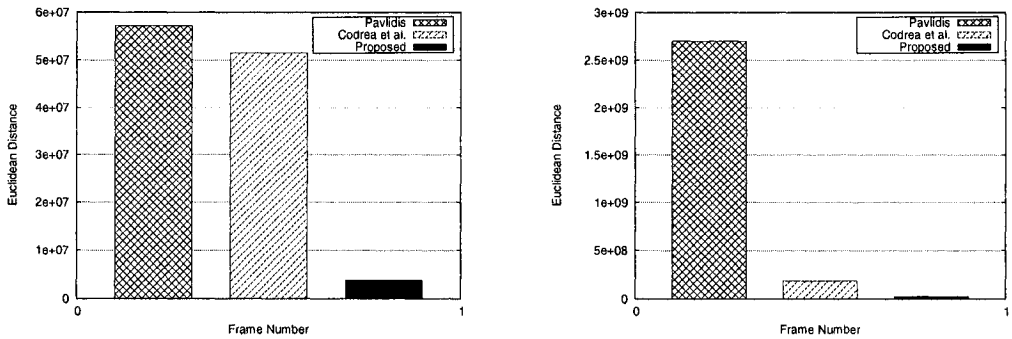
We objectively quantify both the traced and filled contours of each algorithm at the pixel level against the contour ground-truth of that sequence, by means of Euclidean distance [50] defined as

$$\begin{aligned} d^2_{E_C}(C, C_{truth}) &= \sum_{k}^{M}\sum_{l}^{N}(C(k,l) - C_{truth}(k,l))^2, \\ d^2_{E_F}(F, F_{truth}) &= \sum_{k}^{M}\sum_{l}^{N}(F(k,l) - F_{truth}(k,l))^2, \end{aligned} \tag{5.1}$$

where M and N are the width and height of a frame, and for a given tracing scheme, $C(k,l)$ is the contour point at location (k,l), and $C_{truth}(k,l)$ is ground-truth contour point at location (k,l), $F(k,l)$ is the filled contour point at (k,l), and $F_{truth}(k,l)$ is ground-truth point at (k,l). Note that lower values for this measure corresponds to more representative tracing and filling.

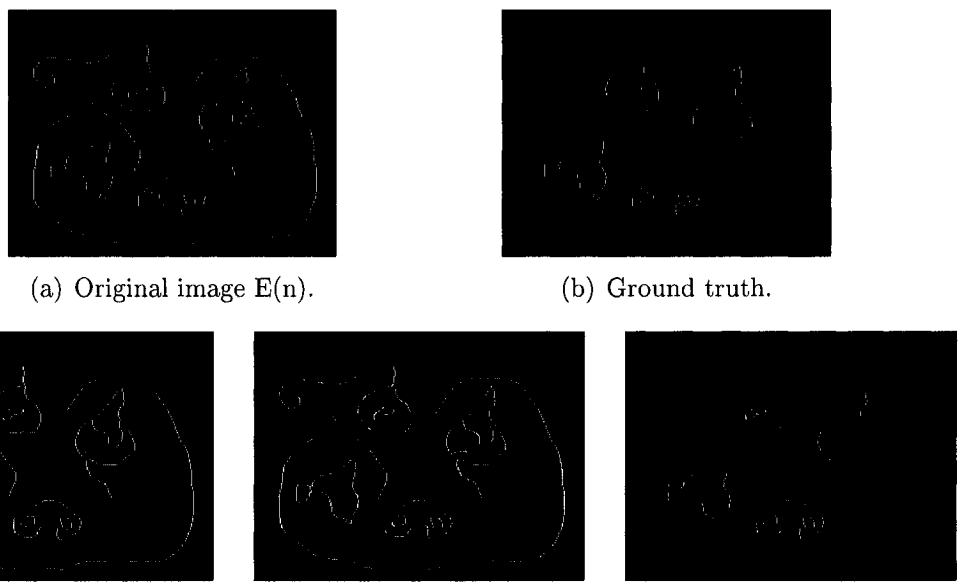
Fig. 5.12 shows the objective comparison for the synthetic image where the output of the proposed tracing and filling schemes better matches the ground-truth as to those of Pavlidis [25, 26] and Codrea et al. [29]. Fig. 5.13 subjective supports Fig. 5.12. Reference tracing algorithms of Pavlidis [25, 26] and Codrea et al. [29] blindly trace the original edge image producing non-representative contours compared to true existing image objects (Figs. 5.13-c,d). Dead or inner branches are added to real contours affecting not only their correspondence to true object but also the extracted features of image contour size, width, height, compactness and irregularity ratios. Inner gap contours are falsely traced as external contours with no correspondence to the containing external contours. Non-closed contours are wrongly considered as valid contours that can greatly effect the performance of subsequent higher object based processing stages.

For "Hall" sequence, the overall advantage and stability of the proposed schemes compared to the reference algorithms is objectively supported by Euclidean distance measures (Fig. 5.14). The subjective evaluation in Fig. 5.15 supports the objective measures. The proposed tracing algorithm is more stable than the reference algo-



(a) Euclidean distance (tracing). (b) Euclidean distance (filling).

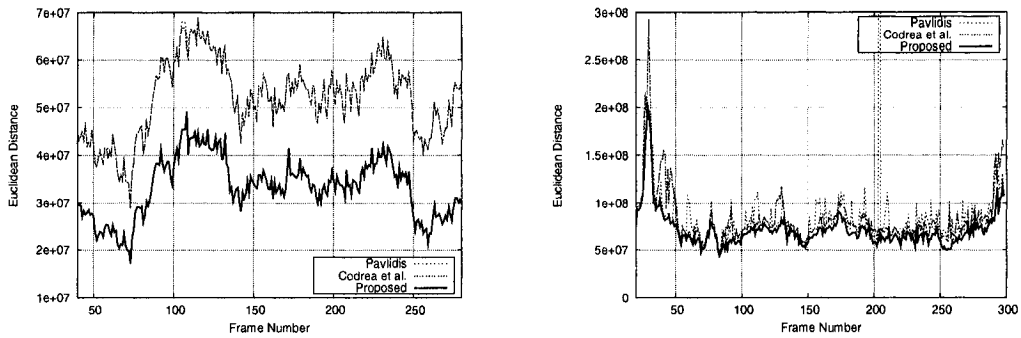
Figure 5.12: Objective comparison of tracing and filling outputs for "Synthetic Image".



(a) Original image E(n). (b) Ground truth.

(c) Pavlidis tracing. (d) Codrea et al. tracing. (e) Proposed tracing.

Figure 5.13: Subjective comparison for "Synthetic Image". For (c), (d) connected closed contours are falsely considered as a single contour, dead or inner branches are falsely added to the real contours, and inner gaps are wrongly traced as separate non-related contours. For (e) dead or inner branches and closed connected contours are correctly detected and rejected, also inner gap contours are labeled correctly (bright white).



(a) Euclidean distance (tracing). (b) Euclidean distance (filling).

Figure 5.14: Objective comparison for "Hall" sequence.

rithms which falsely trace connected closed contours, false inner gap contours and some small noise related contours.

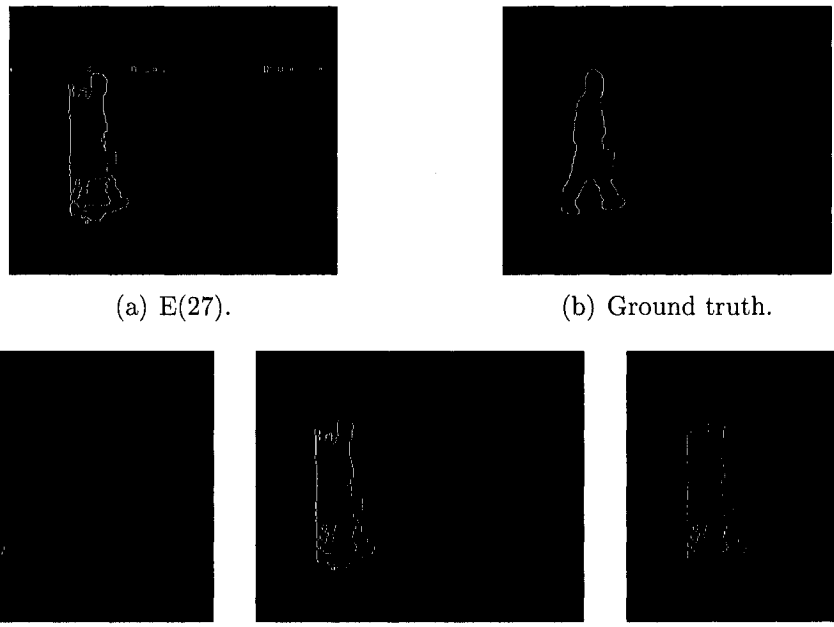
For "Meetingroom" sequence, Fig. 5.16 shows the objective Euclidean distance of tracing and filling results compared to their ground-truth. As can be seen the proposed algorithms clearly outperform the reference algorithms. Fig. 5.17 subjectively supports Fig. 5.16, showing the advantage of the proposed tracing algorithm over the those proposed by Pavlidis [25, 26] and Codrea et al. [29].

5.6.2 Subjective evaluation

For "Hall" Sequence (300 frame of size 352x288):

- The tracing algorithm of Pavlidis [25, 26] produced many problematic frames: i) small dead or inner branches in many frames, ii) small noisy connected closed contours considered as part of the true big contours in various frames (Fig.5.18), iii) tracing failures and filling failures (leaks) in many frames, mainly due to faulty search direction and the in-ability of retracing, or the presence of crossed points. The main filling problem was the wrongly filled inner hole contours (Fig.5.19).

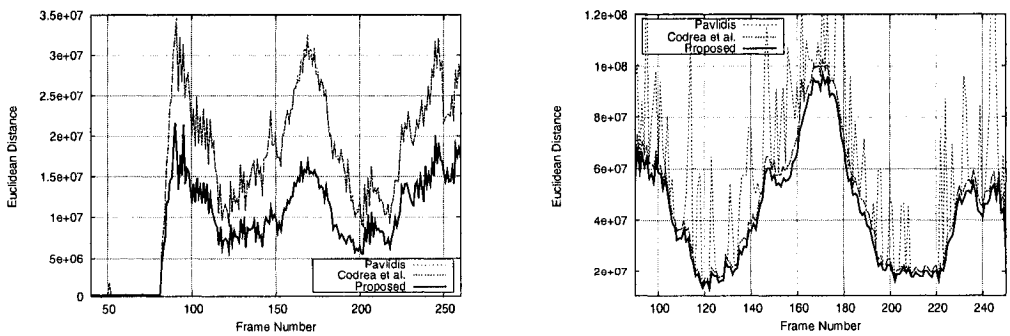
- The algorithms of Codrea et al. [29] has the same tracing problems as those produced by Pavlidis's tracing algorithm (Fig.5.18). Also there was the same filling problem of inner hole contours in many frames (Fig.5.19).



(a) E(27). (b) Ground truth.

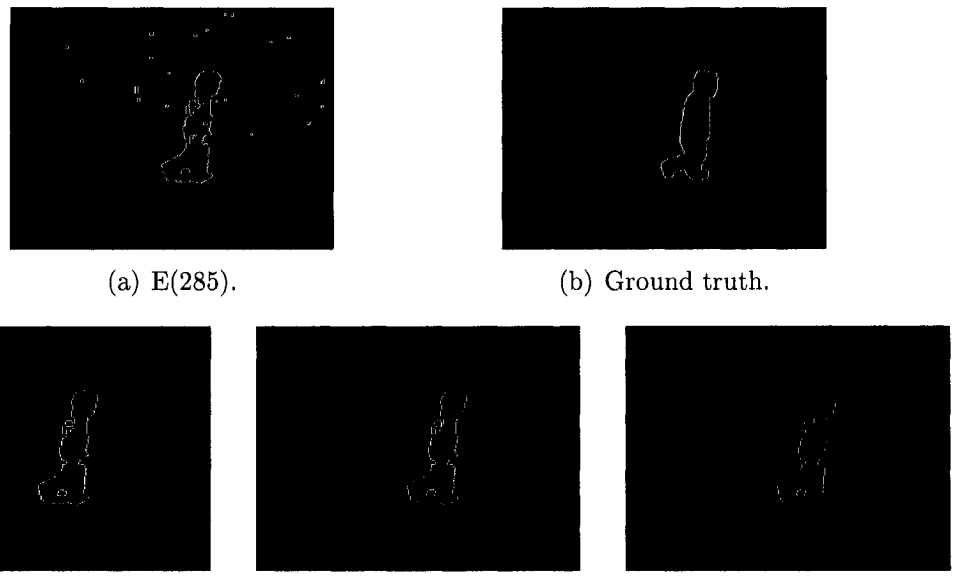
(c) Pavlidis tracing. (d) Codrea et al. tracing. (e) Proposed tracing.

Figure 5.15: Subjective comparison for "HALL" sequence. In (c), (d): the falsely traced and added outer connected closed contour, and the large inner gap contour falsely traced as independent external contour. For (e) the connected closed contour is rejected, and the inner gap contour is detected and labeled correctly (bright white).



(a) Euclidean distance (tracing). (b) Euclidean distance (filling).

Figure 5.16: Objective comparison for "Meetingroom" sequence.



(a) E(285). (b) Ground truth.

(c) Pavlidis tracing. (d) Codrea et al. tracing. (e) Proposed tracing.

Figure 5.17: Subjective comparison for "Meetingroom" sequence. Note in (c), (d) the falsely traced and added connected closed contour and falsely traced inner gap contour as independent external contour. For (e) the connected closed contour is rejected , and the inner gap contour is detected and labeled correctly (bright white).

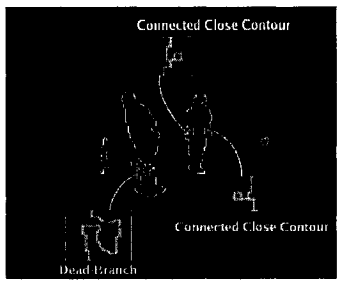
- The proposed set of algorithms succeeded in both tracing and filling all the frames without any tracing or filling failures, and without any of the problematic issues related to object-based video processing.

For "Survey" sequence (1000 frames of size 320x240):

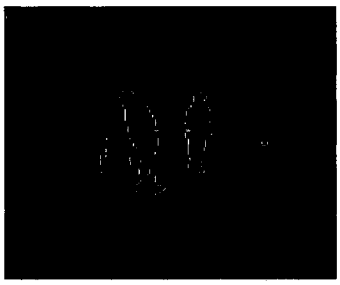
- The algorithms proposed by Pavlidis [25, 26] had more of the same problems as those in the "Hall" and "Highway" sequences (Fig.5.20).

- The Codrea et al. [29] tracing algorithm due to the complexity of this sequence produced more undesirable contours for object-based video processing. More than that there was a noticeable increase in filling errors especially those related to connected inner contours and inner hole contours (Fig.5.20).

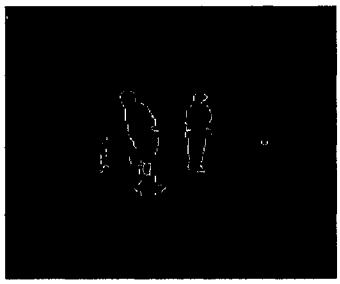
- Proposed set of algorithms also succeeded in both tracing and filling all the frames without any problems.



(a) E(137).



(b) Pavlidis tracing.



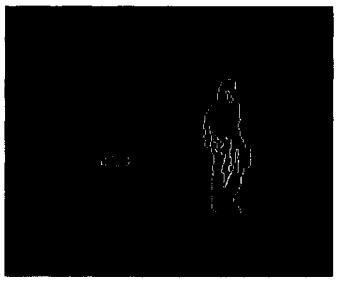
(c) Codrea et al. tracing.

(d) Proposed tracing.

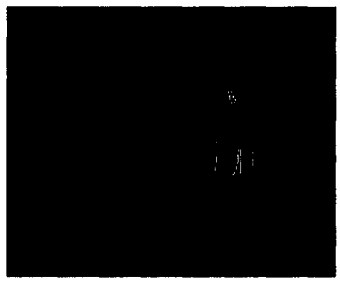
Figure 5.18: "Hall" sequence: non-representative tracing by reference methods of dead or inner branches and connected small contours.

(a) E(291).

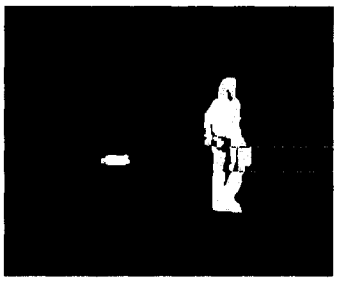
(b) Pavlidis tracing.



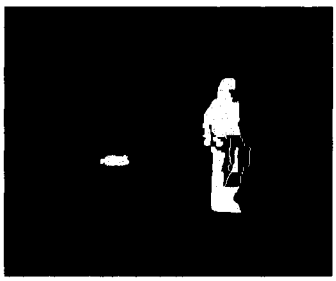
(c) Codrea et al. tracing.



(d) Proposed tracing.



(e) Pavlidis filling.



(f) Codrea et al. filling.

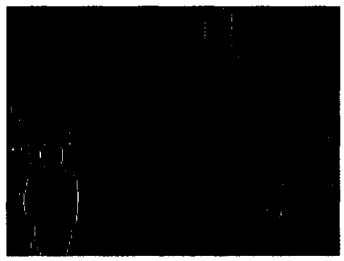
(g) Proposed filling.

Figure 5.19: "Hall" sequence: (b), (c) produce falsely traced multiple inner dead branches and inner gap contours that falsely traced as independent external contours. For (d) all multiple dead inner branches that resulted from initial tracing of the external contour are correctly rejected, while correctly keeping the resulting inner gap contour (bright white). For (e), (f) multiple dead inner branches and inner gap contours caused leaks or non-filled part in the outer containing contour.

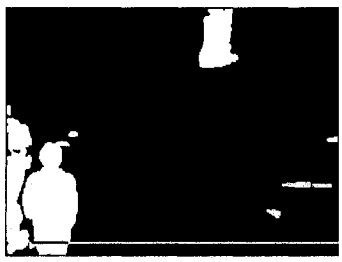
(a) E(6).

(b) Pavlidis tracing.

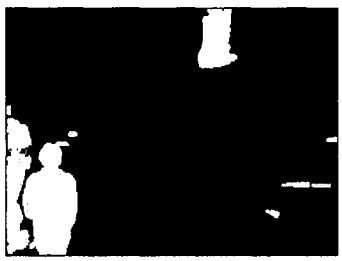
(c) Codrea et al. tracing.



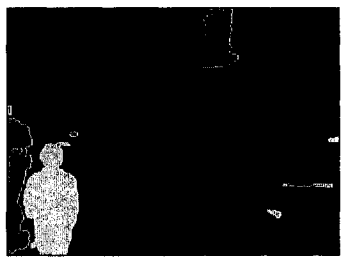
(d) Proposed tracing.



(e) Pavlidis filling.



(f) Codrea et al. filling.



(g) Proposed filling.

Figure 5.20: "Survey" sequence: (b), (c) the contours of the two persons are falsely joined as single contour, the inner gap contour (in left-most person contour) is falsely traced as independent external contour. In (d), the two connected contours are correctly detected and labeled separately (showing as different gray values), the large inner gap contour is detected and labeled correctly. In (e), a filling leak occurred due to the presence of a cross-point. In (f), the inner gap contour of the left-most contour is falsely filled, and leak occurred in the outer containing contour.

5.7 Summary

In this chapter we proposed schemes for contour tracing and filling for object-based video applications. The proposed tracing algorithm locates, in a contra-clockwise manner, all the 8-neighbor connected closed contours. The proposed tracing algorithm that builds on [2], is fundamentally different from classical image based tracing algorithms, since it checks the correctness of video contour shapes at pixel level by

detecting and amending any cases where unwanted distortions (e.g. closed connected contours) may occur at each new candidate contour pixel. The filling algorithm uses the traced contour points and their chain-code information as seed points for horizontal line growing.

The proposed set of tracing and filling algorithms outperformed, both objectively and subjectively, the reference algorithms. This was evident in i) handling complex contours such as dead, inner branches, or crossed contours, ii) avoiding the many noise related small contours, ii) handling connected contours that should not be appended to the original bigger contours as it is the case in all the currently proposed tracing algorithms in literature, but should be addressed as separate closed contours, iii) never re-tracing the same set of contour points twice which greatly reduces tracing time especially in complex video sequences with noise or highly occluded objects, and iv) every extracted contour is stored and processed as an independent object with specific properties which is a very important issue from subsequent video processing stages.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented a set of schemes for robust real-time video segmentation (background subtractions including background update and object detection, and contour tracing and filling). These schemes can be directly and effectively employed in real-world video processing subsystems such as video surveillance, event analysis, and content-based video coding.

We proposed first a novel MOG-based real-time background update technique for moving object detection and foreground mask formation. In this technique, the background update stage uses a new selective matching scheme based on the combined scheme approaches of component ordering and winner-takes-all. This matching scheme not only selects the most probable component for the first matching with new pixel data, greatly improving performance, but also simplifies pixel classification and component replacement in case of no match. Further performance improvement is achieved using a new simple yet functional component variance adaptation formula. The used periodical weight normalization scheme prevents merging temporary stopped real foreground object, and the creation of false ghosts in the foreground mask when these objects start to move again.

The proposed object detection includes, hysteresis-based component matching and temporal motion history schemes. Component hysteresis matching improves

detected foreground (object-binary) blobs by reducing the amount of cracks and added shadows, while motion history preserves the integrity of moving objects boundaries, both with minimum computational overhead.

The proposed background update technique implicitly handles both gradual illumination change and temporal clutter problems. The problem of shadows and ghosts is partially addressed by the proposed hysteresis-based matching scheme. The problem of persistent sudden illumination changes and camera perturbations are addressed at frame level depending the percentage of pixels classified as foreground. We implemented three different state-of-the-art background subtraction techniques and compared their segmentation quality and computational performance with those of the proposed technique.

The proposed background subtraction technique produces binary object pixels that are highly abstract and must be grouped together to form the actual objects. To extract actual objects we proposed contour tracing and filling methods. The improved contour tracing algorithm is fundamentally different from classical still-image oriented tracing algorithms, for it considers video contour representativeness at the pixel level by detecting and correcting any cases where potential contour distortion (e.g. dead branches) may occur at each new candidate contour pixel. The proposed tracing algorithm can detect and reject dead or inner branches, false non-closed contours, noise related small contours, and then efficiently categorize each contour into inner or outer contours. The novel filling algorithm is efficient and never leaks, it uses the extracted contour points and their chain-code information as seed points for horizontal line growing. Experimental results show that the proposed tracing and filling technique improves computational performance with no tracing or filling errors compared to other reference techniques.

6.2 Future Work

There are issues to consider in order to further improve the performance of the proposed techniques. In the background update stage, for any monitored scene, most of

the pixels actually correspond to background pixels where no foreground object ever come into their view. Possible future work would be to consider a different periodical component update scheme *only* for these specific pixels which will greatly improve computational performance. Another future work is to fully address the shadows and ghosts problem at the object detection stage. (Note that we partially address the shadow and ghost problem.) Also for the object detection stage, we think that it is very important to investigate the effect of feedback from higher logical processing stages (e.g., from object tracking) and not rely only on pixel or region level data.

The proposed filling algorithm is used to count pixels inside each video object contour. The counted pixels correspond to the area feature of the traced video object. Possible future work would be to compute the area feature based on video object contour size and chain-code informations, which will greatly improve computational performance.

Bibliography

[1] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. IEEE Conf. Computer Vision (ICCV)*, (Los Alamitos, CA, USA), pp. 255–261, Sept. 20–27 1999.

[2] A. Amer, "Memory-based spatio-temporal real-time object segmentation," in *SPIE Int. Symposium on Electronic Imaging, Conf. on Real-Time Imaging (RTI)*, (Santa Clara, USA), p. 10, 2003.

[3] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *IEEE Workshop on Machine Vision for Intelligent Vehicles (CVPR)*, (Los Alamitos, CA, USA), pp. 2246–2252, 1999.

[4] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," in *Proc. European Workshop. Advanced Video-based Surveillance Systems (ECAVSS)*, (Kingston, UK), pp. 90–95, Sept. 2001.

[5] F. Porikli and O. Tuzel, "Human body tracking by adaptive background models and mean-shift analysis," in *IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS-ICVS)*, (Beijing, China), Mar. 2003.

[6] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, (Cambridge, UK), pp. 28–31, Aug. 2004.

[7] Q. Zang and R. Klette, "Background subtraction using multi-layered mixture model," in *Proc. Int. Conf. Imaging Science, Systems, and Technology (CISST)*, (Las Vegas, Nevada, USA), pp. 88–93, June 2004.

[8] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 5, pp. 827–832, 2005.

[9] X. Luo and S. M. Bhandarkar, "Real-time and robust background updating for video surveillance and monitoring," in *Proc. Int. Conf. Image Analysis and Recognition (ICIAR)*, vol. 3656, (Toronto, Canada), pp. 1226–1233, Sept. 2005.

[10] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *Proc. IEEE Conf. Computer Vision (ICCV)*, (Kerkyra, Greece), pp. 80–97, Sept. 1999.

[11] A. M. McIvor, "Background subtraction techniques," tech. rep., Reveal Ltd, O.O.Box: 128-221, Remuera, Auckland, New Zealand, May 2000.

[12] W. Power, P. Wayne, P. Johann, and A. Schoonees, "Understanding background mixture models for foreground," in *Image and Vision Computing*, pp. 267–271, Dec. 2002.

[13] O. Javed, K. Shafique, and M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information," in *IEEE Workshop on Motion and Video Computing*, (Orlando, FL, USA), pp. 22–27, Dec. 2002.

[14] A. Elgammal, R. Duaiswami, D. Harwood, and L. Davis, "Background and foreground modeling using non-parametric kernel density estimation for visual surveillance," in *Proc. of the IEEE*, vol. 90, pp. 1151–1163, 2002.

[15] E. Hayman and J.-O. Eklundh, "Statistical background subtraction for a mobile observer," in *Proc. IEEE Conf. Computer Vision (ICCV)*, (Beijing, China), pp. 67–74, July 2003.

[16] S. C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," tech. rep., Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550, USA, July 2003.

[17] D. Farin, P. H. de With, and W. Effelsberg, "Robust background estimation for complex video sequences," in *Proc. IEEE Conf. Image Processing (ICIP)*, (Barcelona, Spain), pp. 145–148, Sept. 2003.

[18] Q. Zang and R. Klette, "Robust background subtraction and maintenance," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, (Cambridge, UK), pp. 90–93, Aug. 2004.

[19] M. Piccardi and T. Jan, "Mean-shift background image modelling," in *Proc. IEEE Conf. Image Processing (ICIP)*, (Lion City, Singapor), pp. 3399–3402, Oct. 2004.

[20] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Statistical modeling of complex backgrounds for foreground object detection," *IEEE Trans. Image Processing (IP)*, vol. 13, no. 11, pp. 1459–1472, 2004.

[21] M. Piccardi, "Background subtraction techniques: a review," in *Proc. IEEE Conf. Systems, Man, and Cybernetics (SMC)*, (Hague, Netherlands), pp. 3099–3104, 2004.

[22] S. M. Bhandarkar and X. Luo, "An efficient background updating scheme for real-time traffic monitoring," in *IEEE Trans. Syst.*, (Osaka, Japan), pp. 859–864, Oct. 2004.

[23] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, no. 10, pp. 1337–1342, 2003.

[24] A. M. Elgammal, R. Duraiswami, and L. S. Davis, "Efficient kernel density estimation using the fast gauss transform with applications to color modeling

and tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, no. 11, pp. 1499–1504, 2003.

[25] T. Pavlidis, "Contour filling in raster graphics," in *Proc. ACM Conf. Special Interest Group on Graphics and Interactive Techniques, Computer Graphics (SIGGRAPH)*, vol. 15, (Dallas, Texas, USA), pp. 29–36, Aug. 1981.

[26] T. Pavlidis, *Algorithms for Graphics and Image Processing.* Springer, 1982.

[27] F. Chang and C.-J. Chen, "A component-labeling algorithm using contour tracing technique," in *Proc. Int. Conf. Document Analysis and Recognition (ICDAR)*, (Edinburgh, Scotland), pp. 741–745, Aug. 2003.

[28] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding (CVIU)*, vol. 93, pp. 206–220, Feb. 2004.

[29] M. C. Codrea and O. S. Nevalainen, "Note: An algorithm for contour-based region filling," *Computers and Graphics (CG)*, vol. 29, pp. 441–450, June 2005.

[30] J.-C. Shim and C. Dorai, "A generalized region labeling algorithm for image codingrestorationand segmentation," in *Proc. IEEE Conf. Image Processing (ICIP)*, (Kobe, Japan), pp. 46–50, Oct. 1999.

[31] A. Chehikian, "Image segmentation by contours and regions cooperation," in *Signal Processing*, vol. 78, pp. 329–347, Elsevier Science Publishers, Nov. 1999.

[32] R. T. Collins, A. J. Lipton, and T. Kanade, "A system for video surveillance and monitoring," in *American Nuclear Society 8th Internal Topical Meeting on Robotics and Remote Systems*, (Pittsburgh, Pennsylvania, USA), pp. 110–125, Apr. 1999.

[33] C. K. H. Chiu and K.-Y. Lam, "Supporting real-time active visual surveillance in wireless network," tech. rep., CS department, Hong Kong University, 2003.

[34] S. M. Desa and Q. Salih, "Image subtraction for real time moving object extraction," in *Proc. Int. Conf. Computer Graphics, Imaging and Visualization (ICCGIV)*, (Aachen, Germany), pp. 41–45, Apr. 2004.

[35] R. Cutler and L. S. Davis, "View-based detection and analysis of periodic motion," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, (Brisbane, Australia), pp. 237–242, Aug. 1998.

[36] B. P. Lo and S. A. Velastin, "Automatic congestion detection system for underground platforms," in *Proc. IEEE Synp. Intelligent Multimedia, Video and Speech Processing*, (Kowloon Shangri-La, Hong Kong), pp. 158–161, May 2001.

[37] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: A real time system for detecting and tracking people," in *IEEE Workshop on Machine Vision for Intelligent Vehicles (CVPR)*, (Santa Barbara, CA, USA), p. 962, June 1998.

[38] I. Haritaoglu, D. Harwood, and L. S. Davis, "Hydra: Multiple people detection and tracking using silhouettes," in *Proc. IEEE Conf. Image Analysis and Processing (ICIAP)*, (Venice, Italy), pp. 280–285, 1999.

[39] N. Oliver, B. Rosario, and A. Pentland, "A bayesian computer vision system for modeling human interaction," in *Proc. IEEE Conf. Computer Vision Systems (ICVS)*, (Las Palmas, Spain), pp. 255–272, Jan. 1999.

[40] 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, 1995.

[41] P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, *Video-Based Surveillance Systems: Computer Vision and Distributed Processing.* Kluwer, Nov. 2001.

[42] K.-P. Karmann and A. von Brandt, "Moving object recognition using an adaptive background memory," *Time-Varying Image Processing and Moving Object Recognition, ElsevierScience Publishers B.V.*, vol. 2, pp. 289–307, Mar. 1990.

[43] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in *Proc. European Conf. Computer Vision (ECCV)*, (Stockholm, Sweden), pp. 189–196, May 1994.

[44] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," in *Proc. IEEE Conf. Automatic Face and Gesture Recognition (FG)*, (Killington, Vermont, USA), pp. 51–59, Oct. 1996.

[45] J. Heikkilä and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Proc. IEEE Workshop. Visual Surveillance (VS)*, (Fort Collins, CO, USA), pp. 74–81, IEEE Computer Society, June 1999.

[46] J. Heikkilä and O. Silvén, "A real-time system for monitoring of cyclists and pedestrians," *Image Vision Comput*, vol. 22, no. 7, pp. 563–570, 2004.

[47] T. E. Boult, R. J. Micheals, X. Gao, P. M. Lewis, C. Power, W. Yin, and A. Erkan, "Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets," in *Second IEEE Workshop on Visual Surveillance*, (Fort Collins, CO, USA), pp. 48–55, July 1999.

[48] D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R. B. Fisher, J. S. Victor, and J. L. Crowley, "Comparison of target detection algorithms using adaptive background models," in *Proc. IEEE Workshop. Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, (Beijing, China), pp. 113–120, Oct. 2005.

[49] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust kalman filter," in *Proc. IEEE Conf. Computer Vision (ICCV)*, (Beijing, China), pp. 44–50, 2003.

[50] R. O. Duda, P. E. P. E. Hart, and D. G. Stork, *Pattern classification.* WILEY Publication, second ed., 2001.

[51] C. Su and A. Amer, "A real-time adaptive thresholding for video change detection," in *Proc. IEEE Int. Conference on Image Processing (ICIP)*, (Atlanta, GA, USA), Oct. 2006.

[52] Q. Zang and R. Klette, "Robust background subtraction and maintenance," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, (Cambridge, UK), pp. 90–93, Aug. 2004.

[53] Q. Zang and R. Klette, "Evaluation of an adaptive composite gaussian model in video surveillance," in *Proc. Int. Conf. Computer Analysis of Images and Patterns (CAIP)*, vol. 2756, (Groningen, Netherlands), pp. 165–172, Aug. 2003.

[54] W. H. Press *et al.*, "Numerical recipes in (c) (second edition)," *Cambridge University Press*, 1992.

[55] R. M. Haralick, "Some nearest neighbor operations," in *Real-Time/Parallel Computing: Image Analysis* (M. Onoe, K. Preston, Jr., and A. Rosenfeld, eds.), pp. 11–35, New York, NY, USA: Plenum Press, 1981.

[56] R. Lumia, L. Shapiro, and O. Zuniga, "A new connected components algorithm for virtual memory computers," *Computer Vision, Graphics, and Image Processing*, vol. 22, no. 2, pp. 287–300, 1983.

[57] R. Lumia, L. Shapiro, and O. Zuniga, "A new connected components algorithm for virtual memory computers," *Computer Vision, Graphics Image Processiong (CVGIP)*, vol. 22, pp. 287–300, May 1983.

[58] Y. Shima, T. Murakami, M. Koga, H. Yashiro, and H. Fujisawa, "A high speed algorithm for propogation-type labeling based on block sorting of runs in binary images," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, (Atlantic City, New Jersey, USA), pp. 655–658, July 1990.

[59] C. Fiorio and J. Gustedt, "Two linear time union-find strategies for image processing," *ACM Theoretical Computer Science*, vol. 154, 1996.

[60] R. F. Sproull, *Principles of interactive graphics.* McGraw-Hill, 1979.

[61] J. M. Lane, R. Magedson, and M. Rarick, "An algorithm for filling regions on graphics display devices," *ACM Trans. Graphics*, vol. 2, no. 3, pp. 192–196, 1983.

[62] A. Amer, "New binary morphological operations for effective low-cost boundary detection," *The International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 17, no. 2, pp. 201–214, 2003.

[63] U. Shani, "Filling regions in binary raster images: a graph-theoretic approach," in *Proc. ACM Conf. Special Interest Group on Graphics and Interactive Techniques, Computer Graphics (SIGGRAPH)*, vol. 14, (Seattle, Washington, USA), pp. 321–327, July 1980.

Appendix A

List of Symbols and Abbreviations

A.1 List of Symbols

L	Buffer length of non-recursive techniques
ω_b	Median-filter based adaptive weighting factor
$S^t(I(p))$	A set of buffer intensity values of median-filter based technique
I_{RGB}	Color intensity values of a pixel
$B^t(p)$	background model at frame t
$I_{S,t}(p)$	Predicted background model pixel intensity value for a pixel in a linear predictive filter
a_k	Prediction coefficients in a linear predictive filter
$\eta(\mu_{t,k}, \Sigma_t)$	Kernel Gaussian function of a non-parametric technique
Σ_t	Covariance matrix of color intensity values
ϕ_v	Eigenvectors matrix
E	Identity matrix
ρ	Likelihood of a pixel value

I_t	Current input frame at time t
B	Background model components
BgM_t	Current background model
$p_c \in I_t$	Input pixel at (i, j) location
$I_t(p_c)$	RGB or gray intensity value at p_c
$p_b \in BgM_t$	Background model pixel at (i, j) location
$\{G_{1,t}, \ldots, G_{k,t}\}$	K Gaussian components of each p_b at time instance t
$\mu_{k,t}$	Mean of the $G_{k,t}$ component
$\sigma_{k,t}$	$G_{k,t}$ variance, and assumed to be the same for all RGB color channels
$\omega_{k,t}$	Weight or counter of $G_{k,t}$
$G_{m,t}$	Matched component at time t
λ_1	λ_2 are the hysteresis-based variance thresholds
ω_{gts}	Frame counter (pixel data integration threshold)
α	Parameter adaptation rate
ω_{norm}	Normalization weight threshold
T_{stop}	Sleeping person time threshold
λ_{bf}	Secondary background component threshold
n_{sdn}	Number of p_c that are classified as foreground pixels per new frame
T_{sdn}	Sudden illumination threshold
MH_t	Current motion history frame that is the accumulated result of subsequent temporal differencing

$p_{mh,t} \in MH_t$ Motion history of a pixel at (i, j) location

h_L History length

$ps_{proc} \in BgM_t$ Surrounding processed-pixels of p_c within a defined area

F_{size} Current frame size

B_t Detected object frame (foreground mask)

C_t List of contours in the original video frame I_t at time t,

$C_c \in C_t$ Current contour with starting pixel p_s,

MBB_c Minimum bounding box of C_c,

p_c Current processed pixel,

p_i 8-neighbor of p_c, and p_n corresponds to the next candidate contour pixel of p_c,

d_s Search direction for the next contour pixel candidate

$p_i(d_s)$ 8-neighbor of p_c at location corresponding to d_s

A.2 Abbreviations

ARMA	Autoregressive Moving Average Model
ECE	Electrical and Computer Engineering
KDE	Kernel Density Estimation
LAG	Line Adjacency Graph
LPF	Low Pass Filter
MJPG	Motion JPG
MOG	Mixture of Gaussians
MPEG	Moving Picture Experts Group
PCA	Principle Component Analysis
pdf	Probability Density Function
RGB	Red, Green and Blue color triplet
YUV	A video Coding scheme
LAGs	Line Adjacency Graphs
MBB	Minimum Bounding Box

Appendix B

Algorithms Bench Application

B.1 Description

We developed a universal multi-threaded robust bench application to test both the proposed set of algorithms and the other reference algorithm. We used C++ development environment, thus if recompiled can work under all operating systems. We incorporated in this application the following functionality (Fig. B.1):

- Novel MOG-based segmentation and post processing algorithms
- State of the art segmentation reference algorithms
- Robust multi-threading support
- Real-time simultaneous viewing, capturing and processing from any connected network cameras, web-cams, DV cameras
- Support for YUV and Motion JPG video streams
- Dynamical easy algorithm parameter change

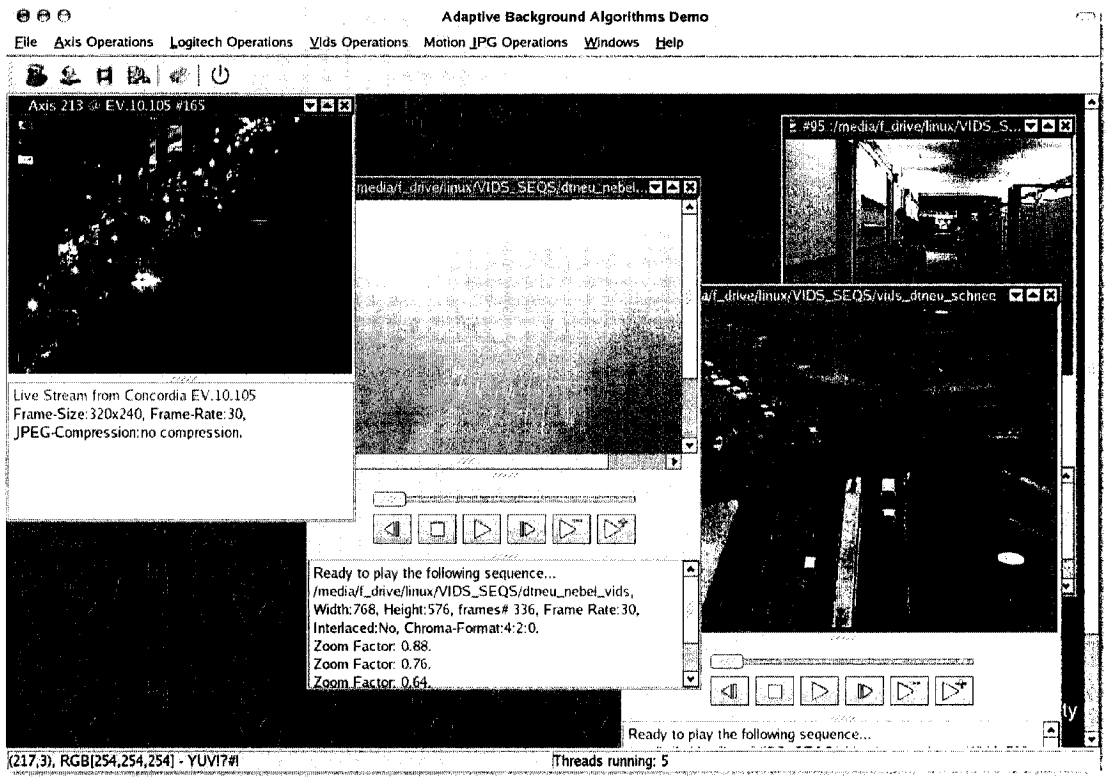


Figure B.1: Screen shot of the test bench application.

B.2 Installation and Usage Guide

If TrollTech QT (www.trolltech.com/products/qt) is installed on a target system, the application is built from the source code by simple 'make'. The produced binary can run directly without the need for installation or any third party libraries.

Following the launch of the application (the GUI) , connected video devices are detected and there corresponding menus are created. Using the application is straight foreword, from each corresponding device menu, a new multi-threaded child-window can be launched either for viewing and capturing streams, or for real-time stream-processing (while viewing). The functionality of each device menu or its sub-menus are clearly described in two forms, tool-tip and status-bar message techniques.

We also provide with the source code, DOXYGEN based documentation system as a development support and coding guidance.

B.3 On-Line System Operation

In the development process of the proposed BS technique, we conducted long hours of real-time simulations on video streams obtained from network cameras for different scenes. In order to evaluate the long term performance of each algorithm, we applied the proposed as well as the reference techniques on on-line captured video scenes from downtown Montreal. These scenes were captured at different time of the day with different weather conditions, including heavy snow, heavy rain, and clouds. On-line simulations were performed also at night and under heavy traffic. The accuracy of the object detections was satisfactory. Note that the scenes included different objects (e.g., pedestrians, cars, buses, bicycles, waving trees, and traffic or advertisement lights.)