Video Object Extraction in Distributed Surveillance Systems

Mohammed Asaad Ghazal

A Doctoral Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

September 24, 2010

Your file Votre référence
ISBN: 978-0-494-71136-1
Our file Notre référence
ISBN: 978-0-494-71136-1

ABSTRACT

Video Object Extraction in Distributed Surveillance Systems

Mohammed Asaad Ghazal

Recently, automated video surveillance and related video processing algorithms have received considerable attention from the research community. Challenges in video surveillance rise from noise, illumination changes, camera motion, splits and occlusions, complex human behavior, and how to manage extracted surveillance information for delivery, archiving, and retrieval. Many video surveillance systems focus on video object extraction, while few focus on both the system architecture and video object extraction. We focus on both and integrate them to produce an end-to-end system and study the challenges associated with building this system.

We propose a scalable, distributed, and real-time video-surveillance system with a novel architecture, indexing, and retrieval. The system consists of three modules: video workstations for processing, control workstations for monitoring, and a server for management and archiving. The proposed system models object features as temporal Gaussians and produces: an 18 frames/second frame-rate for SIF video and static cameras, reduced network and storage usage, and precise retrieval results. It is more scalable and delivers more balanced distributed performance than recent architectures. The first stage of video processing is noise estimation. We propose a method for localizing homogeneity and estimating the additive white Gaussian noise variance, which uses spatially scattered initial seeds and utilizes particle filtering techniques to guide their spatial movement towards homogeneous locations from which the estimation is performed. The noise estimation method reduces the number of measurements required by block-based methods while achieving more accuracy.

Next, we segment video objects using a background subtraction technique. We generate the background model online for static cameras using a mixture of Gaussians background maintenance approach. For moving cameras, we use a global motion estimation method offline to bring neighboring frames into the coordinate system of the current frame and we merge them to produce the background model. We track detected objects using a feature-based object tracking method with improved detection and correction of occlusion and split. We detect occlusion and split through the identification of sudden variations in the spatio-temporal features of objects. To detect splits, we analyze the temporal behavior of split objects to discriminate between errors in segmentation and real separation of objects. Both objective and subjective experimental results show the ability of the proposed algorithm to detect and correct both splits and occlusions of objects.

For the last stage of video processing, we propose a novel method for the detection of vandalism events which is based on a proposed definition for vandal behaviors recorded on surveillance video sequences. We monitor changes inside a restricted site containing vandalism-prone objects and declare vandalism when an object is detected as leaving the site while there is temporally consistent and significant static changes representing damage, given that the site is normally unchanged after use. The proposed method is tested on sequences showing real and simulated vandal behaviors and it achieves a detection rate of 96%. It detects different forms of vandalism such as graffiti and theft.

The proposed end-to-end video surveillance system aims at realizing the potential of video object extraction in automated surveillance and retrieval by focusing on both video object extraction and the management, delivery, and utilization of the extracted information.

Acknowledgments

It is my honor and pleasure to thank the many people who made this thesis possible. I cannot overstate my gratitude to my supervisor Dr. Aishy Amer whose support has been manifold. Her sound advice, encouragement, inspiration, and good teaching have had a major impact on the development of this thesis.

I would like to also thank each member of my supervisory committee for their great comments and questions.

I have truly enjoyed and benefited from my time at the VidPro Research Group. I wish to thank my colleagues at VidPro for their quality work. My work is not possible without theirs. I wish to also thank COEN490 students Mouhib Sarhan, Milos Haravan, and Jeck Chen for their help in implementing and testing the surveillance system. I would also like to acknowledge the help of Dr. Carlos Vázquez in the design and implementation of the occlusion and split handling and vandalism detection algorithms. I am especially grateful to the Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting our work.

Last but not least, no words can express my gratitude to my parents Asaad and Amal Ghazal who raised me, supported me, taught me, and inspired me to be the best I can be. I wish to also thank the rest of my family, my brother Mahmoud, and my sisters Linda, Lamia and Lubna. I am endlessly grateful to my beloved wife Sara and daughter Mayar who shared this journey with me.

Contents

List of Figures

List of Tables

List of Notations

Chapter 2 Symbols

Symbol	Meaning of Symbol
AWGN	Additive white Gaussian noise
F, F_η	Noise-free and noisy images, respectively
η	Added AWGN
$F_\eta(i,j)$	A noisy pixel at spatial coordinates (i,j) in F_η
B_{ij}	A block of pixels in F_η
Ψ_{ij}	Set of spatial coordinates making a block
W	Block size
$\mathbf{x} = (i,j)^T$	An intensity homogeneous location
T	Transpose
t	Iteration number
$p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$	The posterior density of the homogeneous location
R	Number of iterations
$\mathbf{z}_{1:t}$	Homogeneity observations
κ	Normalizing constant
$p(\mathbf{z}_t \mid \mathbf{x}_t)$	The likelihood function
$p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$	The dynamic model
$p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})$	The prior given previous measurements
N	Number of particles
$\{(q_t^{(n)}, \omega_t^{(n)})\}_{n=1}^N$	Particles and their weights
$q_t^{(n)}$	Possible homogeneous location
$\omega_t^{(n)}$	Weight assigned to $q_t^{(n)}$
$\hat{\mathbf{x}}$	Most homogeneous local location found
$\mathcal{N}$	Normal distribution
Σ_D	Covariance matrix of the dynamic model
$\zeta_\mathbf{x}$	Homogeneity at location $\mathbf{x}$
$\mathbf{M}_m^d$	Vectors (indexed by m) in direction d from $\mathbf{x}$ to neibhour
D	Number of directions used
ζ_{max}	Maximum inhomogeneity
ζ_{ref}	target homogeneity
σ_ζ^2	Measurement error variance
$\sigma_{B_{ij}}^2$	Variance of a block
s	Seed location
S	Set of seed locations
Σ_s	Covariance matrix of seed distribution
$\mathbf{P}$	Peak signal to noise ratio
$\mathbf{P}_{\hat{x}_s}$	$\mathbf{P}$ at the $\hat{x}$ found from s
$\mathbf{P}_{init}$	Initial estimate of noise $\mathbf{P}$
Q	Subset of $\{\mathbf{P}_{\hat{x}_s}\}$ with outliers removed
$\mathbf{P}_{Th_1}, \mathbf{P}_{Th_1}$	$\mathbf{P}$ thresholds for outliers rejection
$\mathbf{P}_{ref_1}, \mathbf{P}_{ref_1}$	$\mathbf{P}$ references for outliers rejection
$\hat{\sigma}_\eta^2$	Final estimate of noise

Chapter 3 Symbols

Symbol	Meaning of Symbol
l	Frame number
D_l	Difference frame
F_l	Current frame
G_l	Background model of the current frame
B_l	Binary current frame
D_l^μ	Low-pass filtered difference frame
g_{lim}	Noise adaptive gray-level limit
c	Experimentatl g_{lim} adaptation factor
g_{lim}^c	g_{lim} starting point
σ_η	Standard deviation of AWGN noise
P_η	Peak signal to noise ratio of the AWGN
g_{max}	The maximum possible gray-level
$\mathrm{P}_{D^\mu}^{g_{lim}}$	P of D_l^μ obtained with a given g_{lim}
$\mathrm{MSE}_{D_l^\mu}$	The mean-square error between D_l and D_l^μ
α	g_{max} scaling
g_{lim}^{min}	Lower bound on g_{lim}
Q^o	Sequence of spatial points in the object contour
Q_x^o, Q_y^o	Coordinates of Q^o
$L_{Q_x^o}$	Label for Q^o used for contour filling
F_{l+r}	Neighboring frame to F_l
N_r	Size of frame neighborhood
F_{lr}	compensated neighboring frames
(x_i, y_i)	The location of the i^{th} pixel in F_l
(dx_i, dy_i)	The motion vector of the pixel at (x_i, y_i)
$\mathbf{a}$	Affine motion parameters vector
a_1^l	Affine motion parameter 1 at frame l
$a_1^{l-2\rightarrow l}$	Affine motion parameter 1 between frames $l-2$ and l
E_{DFD}	Displaced frame difference
N	Total number of pixels
s	Power of the E_{DFD}
SAD	Sum of absolute differences
SSD	Sum of square differences

Chapter 3 Symbols (cont.)

Symbol	Meaning of Symbol
Z	Random 3-color vector
z	Observation of Z
k	The class index of a pixel in the MoG model
K	The number classes in the MoG model
$P_r(z\|k)$	The likelihood probability of z corresponding to k
$P_r(k)$	The prior probability of a class k
$P_r(z)$	A scaling factor
$\mu_{k,r}$, $\Sigma_{k,r}$	The mean and covariance of $P_r(z\|k)$
$\mathbf{T}$	The transpose
D	The dimension of Z
I	The identity matrix
ω_k	How many pixels where matched to class k
$F_{lr}^{Red}(x_i, y_i)$	Red components of the pixel
$F_{lr}^{Green}(x_i, y_i)$	Green components of the pixel
$F_{lr}^{Blue}(x_i, y_i)$	Blue components of the pixel
$\mu_{k,r}^{Red}$	The mean of the k^{th} Gaussian model of the red color component
$\mu_{k,r}^{Green}$	The mean of the k^{th} Gaussian model of the green color component
$\mu_{k,r}^{Blue}$	The mean of the k^{th} Gaussian model of the blue color component
λ_1, λ_2	Hysteresis based variance thresholds
$\sigma_{k,r}$	Variance of the k^{th} model
α	The update factor
D^Y, D^U, D^V	Difference frames in Y, U, and V color channels
t_l^{Φ}	Empty-frame binarization threshold
ROC	Region of change
t_l^{r}	ROC block threshold
t_l^{b}	non-ROC block threshold
t	Global binarization threshold

Chapter 4 Symbols

Symbol	Meaning of Symbol
VO	Video object
IO	Image object
$n, n-1$	Current frame number, previous frame number
F_n, F_{n-1}	Current frame, previous frame
i, j, k, l	Indecies, ID numbers, labels of image or video objects, or frames
N_n	Number of image objects in the current frame
N_{n-1}	Number of video objects in the previous frame
$\{\mathrm{IO}_i\}_n$	Set of image objects detected in F_n
$\{\mathrm{VO}_j\}_{n-1}$	Set of video objects from previous frames
z^{-1}	One frame delay
IO_{occ}	IO from the occlusion of two or more VOs
$a(\cdot)$	Operator returning the age of a VO
$\mathrm{T}_{\mathrm{age}}$	Age threshold (minimum age in the scene to be stable)
$match(\cdot, \cdot)$	Operator returning the confidence of a match between IO and VO
$\mathrm{T}_{\mathrm{corr}}$	Correspondence-confidence threshold
MBB	Minimum bounding box
D	One side of the MBB
$r(D)$	Returns true of the side D has changed
$m_p(.), m_c(.)$	Return the previous and current motion, respectively
$o(D)$	Returns 1 or -1 based on which D has outwardly changed
$\mathrm{T}_{\mathrm{chg}}$	Allowable amount of change in the motion of D
$d(\cdot, \cdot)$	The Euclidean distance between two objects
$\mathrm{T}_{\mathrm{dist}}$	Distance threshold used to decide when two objects are close
$p(\mathrm{VO}_l, \mathrm{VO}_k)$	Returns the relative position of VO_k with respect to VO_l
$A(.)$	Returns the area of an object
$\bar{D}$	Opposite side to D
$\widetilde{\mathrm{MBB}}(k, n)$	Estimated MBB of VO_k in F_n
L	The last frame where the object was not occluded
Occ	Set of occlusion event information (i.e., objects, MBBs)
OL_n	Set or list of occlusion events
O_n	The number of occlusion events in the current frame
J, J_M	The first and last index in OL_n of Occs involving IO_{occ}

Chapter 4 Symbols
(cont.)

Symbol	Meaning of Symbol
Occ.MBB(2)	The MBB of the second video object involved in Occ
$\{\mathrm{IO}_l\}_{\mathrm{pspt}}$	A list of possible split image objects
Spl	A split event containing the necessary information to fix it
SL	A list of Spl
S_n	The size of SL
BB_{VO_j}	Bounding box for video object VO_j
lp	Log-polar image
k_1, k_2, K_1, K_2	Indicies and size in the log-polar domain
i, j	Sub-band index and decomposition level for wavelet transform
ϕ, φ	Scaling and wavelet functions, respectively
$W_{\phi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)$	Approximation wavelet coefficients at location m_1 and m_2
$W_{\varphi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)$	Details wavelet coefficients at location m_1 and m_2
IC, ES	Information cost and energy signature functions, respectively
D_B	Bhattacharyya histogram distance
P	Probability of a match
Q	The number of compared points in the boundary of objects
$\mathbf{v}_i^I$	Motion vector inside the object boundary
$\mathbf{v}_i^O$	Motion vector outside the object boundary
δ_i	The difference between $\mathbf{v}_i^I$ and $\mathbf{v}_i^O$
w_i	Measures the reliability of the estimated motion vectors
MDOB	Motion difference along object boundaries

Chapter 5 Symbols

Symbol	Meaning of Symbol
VO_i	Video object with ID i
ϕ	Empty set
R_r	Restricted site with identifier r
I_{r_i}	The event that VO_i is inside or entering R_r
N_{r_i}	Normal interaction between VO_i and R_r
V_{r_i}	Vandalism between VO_i and R_r
M_i^l	Signals that VO_i is moving at time l
K	The number of frames used to establish M_i^l
t_M	Threshold: how much apparent motion constitutes real motion
FR	Frame rate
W	Frame width
L_{r_i}	Signals that VO_i is staying long inside R_r
t_c	The percentage of alarming change in R_r
S_i	The event that VO_i has stayed static for a considerably long time
t_T	Number of frames VO_i is static to be in R_r for a long
A	The event that object area is stable
$\Delta\mathrm{A}_i^l$	The stability of the area of VO_iat time l
Q	Set of area difference signs
E_r	Event of normal behavior
VO_c	Damage object
$\lvert\mathrm{VO}_c\rvert$	Area of damage object
$\mathrm{V}_{r_{ic}}$	Complete vandalism of r by i with c as the result

Chapter 6 Symbols

Symbol	Meaning of Symbol
CIF	Common Intermediate Format
SIF	Source Input Format
MJPEG	Motion Joint Photographic Experts Group
MPEG-4	Moving Picture Experts Group - 4
RTSP	Real-time Streaming Protocol
UDP	User Datagram Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
V_i	Video workstation i
Oj	Control workstation j
S	Surveillance server
$C_x^{V_i}$	Component x in V_i
$B_x^{V_i}y$	Buffer between the x and y components of V_i
s	Video source component index
p	Processing component index
e	Encoding component index
l	Control component index
c	Communications component index
t	Streaming component index
v	Video archiving component index
g	GUI component index
UML	Unified Modeling Language
PTZ	Pan, tilt, and zoom
$Obj1$	Video object 1
$Ev1$	Video event 1
k	Frame number
k_{Obj1}, k_{Ev1}	Object and event frame numbers
T	Index of temporal Gaussians
P	Separation between temporal Gaussians
A^k	Example object feature (area)
μ_T^k, v_T^k	Mean and variance of one Gaussian in T
N	The current number of observations in μ_T^k
R	Set of returned results
TP	True positive
FP	False positive
FN	False negative

Chapter 1

Introduction

Video has become an integral part of many applications such as educational, entertainment, medicine, databases, surveillance, and even wireless applications. Due to increasing security concerns, video surveillance applications and its related video processing algorithms continue to draw the attention of both academia and industry. Developing an automated and fast video surveillance system is a challenging task, but one with many promising applications. Many video surveillance systems either focus on video processing algorithms or system architecture, while few systems address both issues. In this thesis, we develop an end-to-end surveillance system by integrating automated video object extraction in a scalable, distributed, system architecture. The result is a functional distributed surveillance system with fast object and event extraction and indexing that facilitates viewing, archiving, and retrieval of surveillance video and information. Through this integration, the thesis also studies the challenges associated with both the extraction of video objects and behavior information and the delivery and utilization of this information towards the end goal of automated surveillance. Moreover, the resultant system creates a framework for testing video processing algorithms and identifying the bottlenecks of their use.

1.1 Background and objectives

An end-to-end surveillance system has two main tasks. The *first* is to automatically extract objects of interest in a surveillance video sequence. Extracting video objects is a three stage process of: 1) finding video objects; 2) tracking them; and 3) identifying behaviors of interest by those objects. For example, to check if a person abandoned a bag next to a restricted area in an airport. The *second* surveillance system main task is to manage extracted surveillance information (e.g., archiving it for retrieval). These two main tasks must be performed fast. Many algorithms are not fast, operating at a low frame rate (e.g., five frames per second) in the hope that future advances to processing speed will eventually improve the frame rate. Recently, CPU manufacturers are focusing on the number of processors in a CPU instead of faster processors. There are other challenges towards performing these two tasks.

The challenges facing the first task of extracting video objects rise from difficulties in its three stages. Finding objects of interest in a surveillance video is called object segmentation. The objective of segmentation is to classify all pixels in one frame to ones belonging to objects and ones belonging to the background. A fast method to accomplish this is to subtract background pixels and threshold the difference. There are obstacles to this approach. The background frame may become outdated as illumination changes due to the passing of time, noise, or camera movement.

Tracking the objects after they are detected is also challenging. Objects may appear in front of other objects in a scene captured by a single camera, leading to pixels of objects being masked or lost. This is called occlusion. Also due to segmentation errors, one object may appear separated into multiple groups of pixels. This is called split. A tracker must detect these cases and handle them. Also, objects often have similar features. The tracker must not be confused by such situations in order to yield reliable tracking information which can be used to analyze object behaviors and detect events of interest.

The event detection process is difficult because object behavior can be unpredictable. Also, there is a complexity gap between features extracted by humans and quantitative features extracted by an automated algorithm. Humans select which features give them the best semantic message, while automated algorithms extract the semantic message from fixed low-level quantitative features such as motion and spatial features. These low-level features are sometimes insufficient to extract the event reliably. The more high-level the event, the harder it is to define how it appears on video. For example, the automatic detection of vandalism in surveillance video signals is a challenging task because of: 1) the complex and unpredictable nature of a vandalism act and the speed at which it may occur; 2) the underlying difficulty of finding a unique definition for vandalism which may vary based on social contexts and applications; 3) the difficulty in distinguishing between normal and vandal interaction between persons and vandalism-prone objects or sites; and 4) the lack of real and publicly available vandalism test video sequences.

The challenges facing the second main surveillance system task of managing extracted surveillance information are associated with the delivery, transmission, presentation, archiving, and retrieval of the collected information and video. Some of these challenges are: 1) functionally distributing the surveillance system tasks to reduce processing loads; 2) communicating extracted surveillance information and video to reduce network load; 3) displaying extracted surveillance information and video with efficient alerting mechanism to reduce response time; and 4) storing surveillance information and video, to enable retrieving them on demand, with reduced storage requirements.

To achieve our main objective of integrating video object extraction in a distributed architecture and build an end-to-end surveillance system, we must study and address these challenges. We can break the main objective into smaller objectives. First, we design a distributed surveillance system architecture that can easily scale and analyze and tune the performance of this architecture. Second, we develop a video object extraction core for this architecture that includes:

- A segmentation algorithm that can handle noise, changing illumination, and moving cameras
  - To handle noise, we need to model it and estimate it. Block-based noise estimation techniques have shown promise in their accuracy and fast performance, but they still require a full search of the image. We aim at improving the efficiency of block-based noise estimation without losing its accuracy using particle filtering techniques in homogeneity localization.
  - To handle changing illumination we integrate background subtraction with mixture of Gaussians background update. Background subtraction techniques are widely used in segmentation due to their low complexity and accurate segmentation. Their good performance, however, is based on the assumption of a fixed background model. We aim at maintaining their accurate and fast performance on changing illumination using the background update.
  - To handle moving cameras, we use motion compensated local frames to generate a background model of the current frame and use background subtraction for segmentation. This approach is computationally complex. We aim at reducing its complexity while producing comparable accuracy to background subtraction with no camera motion.
- A multiple video object tracking algorithm which can cope with partial and total occlusions and handle object split
  - To perform multiple video object tracking, we use a voting-based feature

  matching object tracking. To cope with partial and total occlusion, we aim to use a two stage approach of detection and handling. Detection is based on identifying sudden changes in the spatio-temporal features of objects and correction is done by separating them.

  - To cope with split we use a similar two stage approach of detection through identifying sudden changes in object features and handling through merging of objects.

- A vandalism detection algorithm that combines both the detection of abnormal behavior of vandals and the detection of the vandalism damage.

1.2 Motivation

A reliable and fast automated surveillance system has many applications. Examples are automated security and traffic monitoring. Some security application include: 1) intrusion detection for restricted areas; 2) vandalism detection, which we use in our work as an application; 3) detecting abandoned objects in public transportation areas such as airports and bus stops; and 4) home surveillance, e.g., detecting if a child is playing in a dangerous area of the house. A fast surveillance system over IP can be accessed from mobile devices, thus enabling surveillance from anywhere. For example, when an alert is sent to a mobile device, a security personnel who is away from the security monitors has immediate access to the surveillance feed and to archived information and can act faster. Another area of applications is traffic monitoring. Some of the traffic monitoring applications are: 1) detecting traffic violations such as a speeding car or driving in the wrong direction; and 2) detecting traffic congestion.

Note that research and improved performance in automated surveillance systems requires researching and improving the performance of used video processing algorithms such as noise estimation, video object segmentation, tracking, and event detection, which have many applications of their own. Therefore, the benefits of improving the performance of these video processing algorithms extends beyond their application to video surveillance in this thesis.

Fast and accurate noise estimation is needed in many video processing algorithms such as noise reduction, edge detection, global motion estimation, video coding, and video object segmentation. For example, inaccurate noise estimates in noise reduction lead to blurring of high structure areas, which leads to missing important edges in edge detection. Inaccurate noise estimation also leads to inaccurate global motion parameters increasing the mean absolute error in compensated frames, which leads to needing more bits to code these errors.

Video object segmentation is needed by many video processing algorithms, some of which include video coding, video indexing and retrieval, and video authoring and editing. Many of these algorithms aim to operate in real-time under varying real-world conditions. Therefore, there is a major interest in fast and accurate object segmentation techniques.

Video object tracking is the core of video surveillance applications. Another application of tracking is supporting segmentation. Segmentation is affected by erroneous object merging (i.e., occlusion) and splitting (i.e., fragmentation). Reliable tracking provides a feedback loop to segmentation allowing it to fix those errors. Tracking also facilitates high-level behavior analysis such as vandalism detection. It is also used in robotics and in face detection and recognition.

Vandalism is a major problem. According to the Canadian Centre of Justice Statistics, vandalism in 2004 accounted for up to 36% of all reported crimes [11]. In the year 1991, over two million incidents of vandalism against private property have occurred in the UK [12]. The cost of vandalism is not only financial, but also social. According to a poll for *The Times*, vandalism is regarded by people as one of the most important problems facing their families [12]. The deployment of intelligent video surveillance systems able to detect and report vandalism as it happens is, therefore, becoming popular. There is a growing trend of fitting public transport vehicles (e.g., buses, trains and taxis) with surveillance cameras and eventually having a city-wide surveillance network for detecting vandalism [13].

1.3 Contributions

Research in video surveillance has grown rapidly and a huge number of publications exist. The following list highlights the original parts of the thesis to the best knowledge of the author.

- A new scalable, distributed and real-time surveillance system with indexing and retrieval of surveillance information. The proposed system can be used to realize many surveillance applications. The novelty here is three fold: 1) a scalable system architecture with extensible, and reliable distributed modules; 2) real-time indexing and retrieval of object- and event-based surveillance information and video; and 3) reduced network and storage requirements through reduction of features sampled over time.

- A new method for homogeneity localization using particle filters with application to noise estimation. The proposed noise estimation method reduces the number of block homogeneity measurements needed by block-based approaches by eliminating the need for a full search on the noisy image. It starts by defining uniformly distributed scattered initial seeds and then uses particle filtering techniques to guide those seeds towards nearby homogeneous areas. The novelty in this part is in: 1) reducing the number of homogeneity measurements in block-based techniques while retaining or exceeding their accurate performance in different noise-levels; 2) proposing a dynamic model and a Laplacian-based homogeneity observation model for particle filtering to allow blocks to move towards nearby homogeneous areas; and 3) proposing an adaptive robust estimator to compensate for the reduction in the number of estimation blocks.

- A new algorithm for segmentation under global motion. The novelty in this part is in: 1) a new method for background generation using segmentation-oriented global motion estimation and a mixture of Gaussians based motion-compensated frames merging, which brings significant reduction in computations compared to a recent method; and 2) a new segmentation method using the generated background models from the previous step, which combines color-based object segmentation and morphological post-processing to retain the quality of the recent method while being less complex.

- A new algorithm for the real-time detection and correction of occlusion and split in object tracking for surveillance applications. The novelty in this parts is in: 1) a new method for analyzing the spatio-temporal features of objects to detect sudden variations indicating a possible occlusion; 2) a new method for correcting multiple occlusions by systematically separating occluded objects; and 3) a new method for split detection, which in addition to the analysis of spatio-temporal changes in objects features, analyzes the temporal behavior of split objects to discriminate between errors in segmentation and real separation of objects, such as in a deposit event.

- A new method for vandalism detection in surveillance video sequences by monitoring and evaluating changes inside predefined restricted sites as objects enter or exit these sites. The novelty here lies in a new fast approach to vandalism detection, which does not require training, can handle different forms of vandalism such as damage, graffiti, and theft.

Some of these contributions appear in the research papers [14–18].

1.4 Thesis Outline

The thesis is divided into five chapters each dealing with a specific objective towards building the surveillance system. An overview of the proposed system linking the different chapters together is shown in Fig. 1.1. In Chapter 2, we present the particle filtering based homogeneity localization method and its application to additive white Gaussian noise variance estimation. In Chapter 3, we zoom into a major component of the surveillance system, i.e., segmentation, and present the two methods used for fast segmentation with fixed camera, and segmentation with global motion. In Chapter 4, we present the proposed tracking method with occlusion and split detection and correction. We present the proposed vandalism detection method in Chapter 5. Finally, we present in Chapter 6 the proposed surveillance system as a whole while focusing on the system architecture. Chapter 7 concludes the thesis.

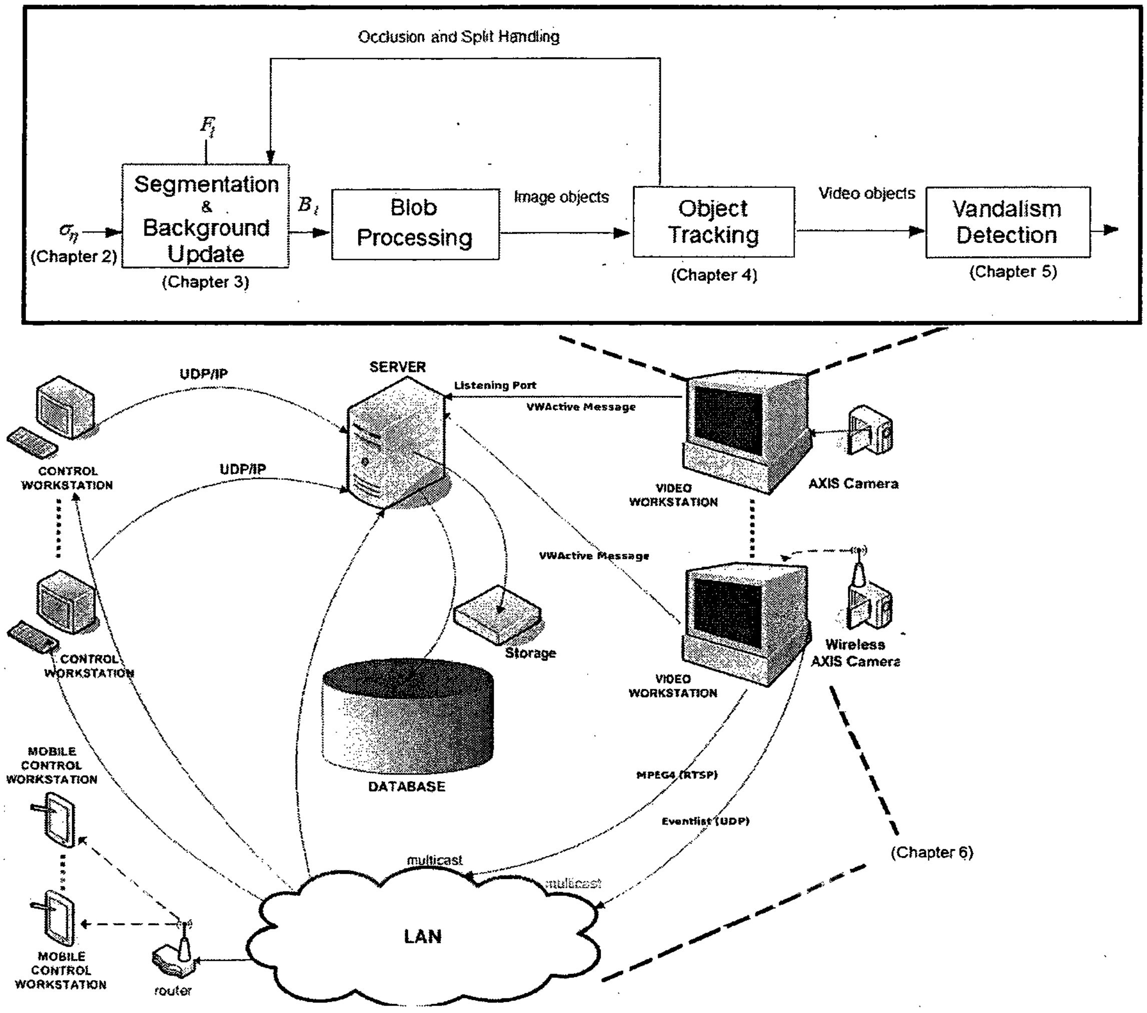


Figure 1.1: Overview of the proposed system linking the chapters together. σ_n is the estimated noise standard deviation. F_l is the current frame and B_l is the binary frame.

Chapter 2

Homgeneity Localization in Noise Estimation

2.1 Introduction

This chapter proposes a method for localizing homogeneity and estimating additive white Gaussian noise (AWGN) variance in images. The proposed method uses spatially and sparsely scattered initial seeds and utilizes particle filtering techniques to guide their spatial movement towards homogeneous locations. This way, the proposed method avoids the need to perform the full search associated with block-based noise estimation methods. To achieve this, the chapter proposes for the particle filter a dynamic model and a homogeneity observation model based on Laplacian structure detectors. The variance of AWGN is robustly estimated from the variances of blocks in the detected homogeneous areas. A proposed adaptive trimmed-mean based robust estimator is used to account for the reduction in estimation samples from the full search approach. The performance measures for noise estimation algorithms are accuracy and computational complexity. Our results show that the proposed method reduces the number of homogeneity measurements required by block-based methods while achieving exceeding or retaining their accuracy.

2.2 Review of Related Work

Accurate estimation of noise is important to many image and video processing algorithms such as motion estimation, edge detection, object segmentation, and denoising. These algorithms are able to adapt their techniques to the noise-level to achieve better performance. Homogeneity localization or measurement is key for accurate and reliable estimation of noise to prevent outliers due to structure from affecting the estimation process.

While there are algorithms, e.g. [19], that consider the effect of non-additive noise which depends on the intensity level, in this thesis, we assume the common an additive white Gaussian noise (AWGN) [20] and accordingly categorize algorithms for estimating the noise variance to temporal [21, 22] or spatial [1–4, 23–29]. Temporal methods have to deal with object or global motion using motion detection or motion compensation leading them to be more computationally expensive than spatial methods. Spatial methods [1–4, 23–29] are challenged by high or low noise levels and images that are dominated by structure. These algorithms seek to strict estimation to regions with the lowest intensity variations. They can be further categorized into smoothing-based, wavelet-based, and block-based methods. Smoothing-based algorithms such as [23] estimate noise from the difference of the noisy and smoothed signal, the difference being an approximation of the noise. The algorithm in [23] limits estimation to homogeneous areas by thresholding the image gradient, but tend to overestimate the noise variance in low noise signals.

Wavelet-based methods [3, 24, 25] use the wavelet transforms to isolate the noise in the diagonal band coefficients and robustly estimate the noise variance from the absolute value of those coefficients. Wavelet based methods deliver more accurate estimation for noisy or high structure images than block-based methods, but are less accurate in low noise levels or low structure images. A similar technique to wavelet methods is used in [4, 26] in which Laplacian-based pseudo residuals are calculated to measure homogeneity and a robust least median of squares or least trimmed of square estimator is used to obtain the noise variance. This approach behaves similar to wavelet-based methods, i.e., is more accurate when noise is the most dominant element in the signal.

Block-based methods such as [1,2,28,29] locate homogeneous regions and estimate the noise from blocks in those regions. They vary in how they detect homogeneous regions. The method in [28] extracts vertical and horizontal detail components and uses histogram information for noise estimation, but is more computationally expensive due to the used optimization. The method in [29] improves upon [28] and uses the Laplacian operator to detect homogeneity. The algorithm in [1] uses elements from [23] and [30]. It uses the minimum block variances to locate homogeneity and estimates noise from the difference between the noisy and smoothed signals over homogeneous blocks. The variance is not always a reliable measure of homogeneity, especially in high noise levels. The performance of [1] depends on the amount of smoothing performed. The algorithm in [2] uses a Laplacian-based homogeneity test composed of a number of high-pass directional operators. The variance of the noise is estimated from the local variances of the blocks selected to be the most homogeneous. The methods in [1, 2, 28, 29] require a full search of the image to locate homogeneous areas. The full search allows these methods to observe homogeneity in all areas and have higher probability of locating the most homogeneous regions. Nevertheless, they also find non-homogeneous areas due to the influence of noise on homogeneity measurements. Computations can be reduced by establishing spatial separation between blocks at the expense of a loss in accuracy due to the loss in search resolution.

The proposed method reduces the number of block homogeneity measurements needed by block-based approaches by eliminating the need for a full search on the noisy image. Reducing the number of homogeneity measurements reduces the complexity of the algorithm. The aim is to reduce the complexity to a level where real-time and accurate noise estimation is achieved. It starts by defining uniformly distributed scattered initial seeds and then uses particle filtering techniques to guide those seeds towards nearby homogeneous areas. The contributions of the proposed method are: reducing the number of homogeneity measurements in block-based techniques while retaining or exceeding their accurate performance in different noise-levels; proposing a dynamic model and a Laplacian-based homogeneity observation model for particle filtering to allow blocks to move towards nearby homogeneous areas; and proposing an adaptive robust estimator to compensate for the reduction in the number of estimation blocks.

The remainder of the chapter is as follows. Sections 2.3 presents the proposed approach. Results are discussed in Section 2.4, and Section 2.5 summarizes the chapter..

2.3 Proposed Algorithm

We start by presenting our framework for locating homogeneous areas using particle filtering. We then move to present how we utilize multiple seeds and use their associated final homogeneous locations to collect estimation samples (i.e., block variances) and robustly use these samples to find the final estimate.

2.3.1 Locating homogeneous areas using a particle filter

Let F_η be a noisy image defined as

$$F_\eta = F + \eta, \tag{2.1}$$

where F is the original noise-free signal and η is the added AWGN component. We denote a single pixel in F_η by $F_\eta(i,j)$ where i and j are the spatial coordinates. We extract blocks from F_η in the estimation process, denoted B_{ij}, using

$$\begin{aligned} B_{ij} &= \{F_\eta(i,j) | (i,j) \in \Psi_{ij}\}; \\ \Psi_{ij} &= \{(i,j) | i - \tfrac{W-1}{2} \leq i \leq i + \tfrac{W-1}{2}, \\ &\qquad j - \tfrac{W-1}{2} \leq j \leq j + \tfrac{W-1}{2}\}, \end{aligned} \tag{2.2}$$

where Ψ_{ij} is the set of spatial coordinates making the block of size W^2 (W is odd) surrounding the center 2D point $(i,j) \in F_\eta$, and B_{ij} is the set of pixel values at those coordinates. Obtaining a reliable estimate of the noise variance σ_η^2 requires finding the set of spatial locations most homogeneous in terms of intensity. Let $\mathbf{x} = (i,j)^T$ be one of those intensity homogeneous locations (T is the transpose). Limiting the estimation to homogeneous areas is important to prevent outliers caused by spatial structure from affecting the estimation process.

We can regard localizing homogeneity as the estimation of the system state using a sequence of noisy measurements made on the system. Our goal is to recursively find the posterior density $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ of the homogeneous location $\mathbf{x}_t$ conditioned on a series of homogeneity observations $\mathbf{z}_{1:t} = (\mathbf{z}_1, .., \mathbf{z}_t)$, where t is the iteration number (we perform R iterations). $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ is calculated recursively from incoming observations in two phases: a prediction phase followed by an update phase. When modeling the spatially-varying state $\mathbf{x}_t$ as a first-order Markov process, $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ becomes,

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \kappa \cdot p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}), \tag{2.3}$$

where κ is a normalizing constant and $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ is calculated using

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}, \tag{2.4}$$

where $p(\mathbf{z}_t|\mathbf{x}_t)$ is the likelihood function, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ dictates the dynamic model and $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ is the prior given previous measurements.

Particle filtering implements, using Monte Carlo simulations, a recursive Bayesian filter [31,32] and have been widely used recently for visual tracking in different applications [33,34]. The posterior density $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ is approximated by a set of N particles and their weights, or $\{(q_t^{(n)}, \omega_t^{(n)})\}_{n=1}^N$, where $q_t^{(n)}$ is a possible homogeneity location and $\omega_t^{(n)}$ is the weight assigned to it as a discrete sample of the $p(\mathbf{z}_t|\mathbf{x}_t = q_t^{(n)})$. We can now estimate the state at each iteration t from $\{(q_t^{(n)}, \omega_t^{(n)})\}_{n=1}^N$, which is propagated according to the system dynamic (see (2.6)) model over the iterations.

The particle filtering algorithm can be summarized as follows:

From $\{(q_{t-1}^{(n)}, \omega_t^{(n)})\}_{n=1}^N$ at iteration $t-1$ $(1 \leq t \leq R)$:

1. **Resample** $\{(q_{t-1}^{(n)}, \omega_{t-1}^{(n)})\}_{n=1}^N$ to produce $\{(q_{t-1}^{\prime(n)}, \frac{1}{N})\}_{n=1}^N$ [35].
2. **Dynamically propagate** each $q_{t-1}^{\prime(n)}$ by the dynamic model $p(\mathbf{x}|\mathbf{x}_{t-1} = q_{t-1}^{\prime(n)})$ and obtain $\{(q_t^{(n)}, \frac{1}{N})\}_{n=1}^N$. Propagation is governed by $x_t = f(x_{t-1}) + v$, where $f()$ is the dynamic or prediction model, and v is Gaussian noise.
3. **Weight** each $q_t^{(n)}$ by $p(\mathbf{z}_t|\mathbf{x}_t = q_t^{(n)})$ to obtain $\{(q_t^{(n)}, \omega_t^{(n)})\}_{n=1}^N$. Weighting is governed by $z_t = h(x_t) + g$. where $h()$ is the observation function and g is the measurement noise.
4. **Find** the most homogeneous location using

$$\hat{\mathbf{x}} = \underset{\omega_t^{(n)}}{\text{argmin}}\ (q_t^{(n)}) \tag{2.5}$$

Algorithm 1: Homogeneity localization using particle filtering

Dynamic Model of the Particle Filter

In essence, the proposed method drives the particles towards homogeneous areas in the image and away from structure guided by a proposed system dynamics and homogeneity observation models. We use the system dynamics model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ (describing the transition from x_{t-1} to x_t) given by

$$p(\mathbf{x}_t|\mathbf{x}_{0:t-1}) = \mathcal{N}(x_t; 2x_t - x_{t-1}, \Sigma_D), \tag{2.6}$$

where $\mathcal{N}$ stands for Gaussian distribution, and Σ_D is its covariance matrix. Equation 2.6 predicts the current location $\mathbf{x}_t$ as the sum of the current location, the displacement $\mathbf{x}_t - \mathbf{x}_{t-1}$ and a zero mean Gaussian noise. This way, the dynamic model pushes particles in the direction where homogeneity was found in the previous iteration.

Observation Model for Homogeneity

We use an observation model based on [2]. Let $\zeta_\mathbf{x}$ be the homogeneity at location $\mathbf{x}$ calculated using

$$\zeta_\mathbf{x} = \frac{1}{D \times \zeta_{max}} \sum_{d=1}^{D} |(W-1)F_\eta(\mathbf{x}) - \sum_{m=1}^{W-1} F_\eta(\mathbf{x} + \mathbf{M}_m^d)|, \tag{2.7}$$

where $\mathbf{M}_m^d$ is a set of $W-1$ vectors (indexed by m) in the direction d starting from $\mathbf{x}$ and ending in a neibhouring location, and D is the number of directions used. We use eight directions specified using

$$\begin{aligned}
\mathbf{M}_m^1 &= \{(0,-1),(0,-2),(0,1),(0,2)\};\\
\mathbf{M}_m^2 &= \{(-1,0),(-2,0),(1,0),(2,0)\};\\
\mathbf{M}_m^3 &= \{(-2,-2),(-1,-1),(1,1),(2,2)\};\\
\mathbf{M}_m^4 &= \{(-2,2),(1,-1),(1,-1),(2,-2)\};\\
\mathbf{M}_m^5 &= \{(-1,0),(-2,0),(0,1),(0,2)\};\\
\mathbf{M}_m^6 &= \{(-1,0),(-2,0),(0,-1),(0,-2)\};\\
\mathbf{M}_m^7 &= \{(1,0),(2,0),(0,1),(0,2)\};\\
\mathbf{M}_m^8 &= \{(1,0),(2,0),(0,-1),(0,-2)\}, \qquad (2.8)
\end{aligned}$$

These directions are the horizontal, vertical, two diagonals and four corner directions. Note that in (2.7) we take the average of directional homogeneity by multiplying with $\frac{1}{D}$ and normalize our homogeneity measure by multiplying with $\frac{1}{\zeta_{max}}$, where $\zeta_{max} = (W-1) \times 255$, where 255 is assumed to be the maximum gray level in the image.

Our observation model is described by

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\zeta_{\mathbf{x}}; (\zeta_{ref} - \zeta_{\mathbf{x}})^2, \sigma_\zeta^2), \qquad (2.9)$$

where ζ_{ref} is our *target* homogeneity, i.e., $\zeta_{ref} = 0$, and $\sigma_\zeta^2 = 10^{-3}$ is the measurement error variance. With noise in the image, it is not possible to reach the target homogeneity of $\zeta_{ref} = 0$, however, it drives the particles towards the most homogeneous areas (closest to ζ_{ref}).

After R iterations, we calculate our block mean and variance from the final location

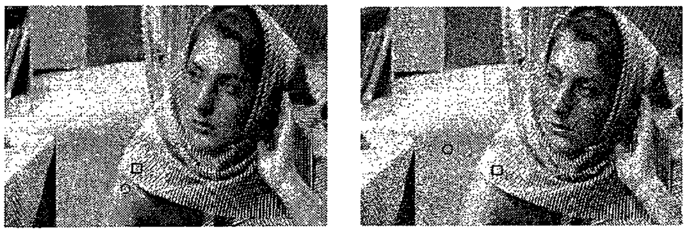
$\hat{\mathbf{x}}$, or location $(i,j) \in \Psi_{ij}$ of (2.2) using

$$\begin{aligned} \mu_{B_{ij}} &= \frac{\sum_{(i,j)\in\Psi_{ij}} F_\eta(i,j)}{W^2}, \\ \sigma^2_{B_{ij}} &= \frac{\sum_{(i,j)\in\Psi_{ij}} (F_\eta(i,j)-\mu_{B_{ij}})^2}{W^2-1}. \end{aligned} \tag{2.10}$$

The variance $\sigma^2_{B_{ij}}$ in a homogeneous region is an estimate of the noise variance.

The block variance $\sigma^2_{B_{ij}}$ can also be used as an alternative approach to measuring homogeneity. When using the block variance to measure homogeneity, we set $\zeta_{\mathbf{x}} = \sigma_{B_{ij}}/255$ in (2.9) and keep $\zeta_{ref} = 0$. This variance-based homogeneity measurement requires more computations than the the Laplacian-based homogeneity measurement.

Starting from a random point (or seed) in the image, the proposed method uses particle filtering to either: 1) move away from structure towards homogeneity if the seed is in a structured area; or 2) stay in the homogeneous areas and not move towards structure if the seed is in a homogeneous area. We can observe the move to homogeneous areas in different noise levels in Fig. 2.1. The square in Fig. 2.1(a) and (b) represent the starting points (or seed) in a structured area for 30 dB and 25 dB noise, respectively. The circles represents the final locations after the end of iterations.



(a) 30 dB noise (b) 25 dB noise

Figure 2.1: Detected homogeneous location (circle) starting from a seed in a structured areas (square) in different noise levels.

2.3.2 Application to Noise Estimation

If we seed the proposed method to localize homogeneity with a location surrounded by structure, it attempts to move away towards a nearby homogenouous location. If we seed it with a location surrounded by homogeneity, it attempts to remain in that location until the end of iterations. Some images are highly structured. The number of iterations used may not be enough to move a seed in the middle of a large structured area towards homogeneity. Moreover, one sample of the block variance as in (2.10) is not enough to yield a reliable estimate of the noise variance.

To reduce this problem, we use a set S of seeds uniformly distributed (see Fig. 2.2) over the image. An alternative approach is to use the posterior knowledge (final seed locations) from one video frame in case of video signals as the prior knowledge (initial seed locations) for the next frame. This is, however, problematic in case of moving objects or cameras as it can potentially increase the distance the blocks have to move to reach homogeneity. Thus, using uniformly distributed seeds is more reliable since the same number of iterations is performed for both cases. Each seed $s \in S$ is modeled as

$$p(s) = \mathcal{N}(s, \Sigma_S). \tag{2.11}$$

We then use the set of final locations from each seed $\{\hat{x}_s\}$ and calculate the block variance around them using (2.10) to produce the set of candidate estimation blocks $\{\sigma^2_{\hat{x}_s}\}$. We also calculate the peak signal to noise ratio (denoted $\mathbf{P}$) from each variance to produce $\{\mathbf{P}_{\hat{x}_s}\}$.

Finally, we use a robust estimator on $\{\sigma^2_{\hat{x}_s}\}$ to find the final estimate $\hat{\sigma}^2_\eta$. The robust estimator is based on the trimmed mean. For trimming, we need to define a threshold and a reference estimate so we can average estimates which do not deviate from the reference estimate more than the threshold. We define the reference estimate and the threshold in the $\mathbf{P}$ domain, this way the threshold when converted back to the

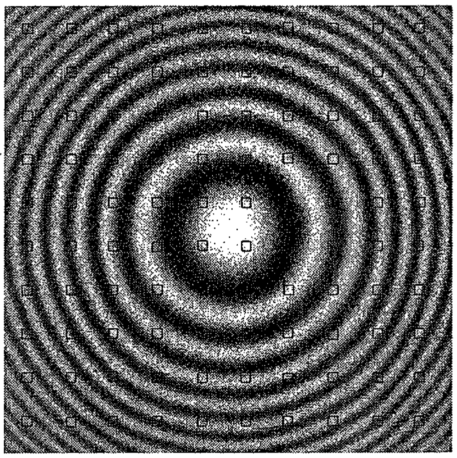


Figure 2.2: The initial seeds evenly distributed on the image. Using random seeds does not improve the performance for noise estimation.

variance domain, naturally increases for noisy signals and decreases for less noisy ones. In [2], a large number of candidate estimates are used, hence the natural decrease in the peak signal to noise ratio threshold (i.e., 3 dB) is sufficient to produce a reliable final estimate for all noise levels. In the proposed method, the number of candidate blocks is reduced significantly and the 3 dB threshold can lead to over estimation of noise for less noisy images. Therefore, we adapt it to an initial estimate of noise $\mathbf{P}_{init}$ by decreasing it to 1 dB for less noisy images (e.g., $\mathbf{P}_{init} > 33$ dB). We use a reference estimate based on the one in [2], i.e., the median of the three least variances. However, we also adapt the reference estimate to $\mathbf{P}_{init}$ due to the significant decrease in the candidate estimates. In noisier images (e.g., $\mathbf{P}_{init} < 33$ dB), underestimates of noise exist due to operating in small blocks and not having enough samples to reflect the true value of noise. Therefore, for noisier images we use the median of the least 10% candidate variances.

For $\mathbf{P}_{init}$, we use the peak signal to noise ratio of the seed with the minimum variance

$$\mathbf{P}_{init} = 10 \log \left(\frac{255^2}{\min(\{\sigma^2_{\hat{x}_s}\})} \right), \tag{2.12}$$

With $\mathbf{P}_{init}$ calculated, we define our robust estimator as one which selects a subset Q of $\{\mathbf{P}_{\hat{x}_s}\}$ (by removing outliers) with

$$Q = \begin{cases} \{\mathbf{P}_{\hat{x}_s} : |\mathbf{P}_{\hat{x}_s} - \mathbf{P}_{ref_1}| < \mathbf{P}_{Th_1}\} & : \mathbf{P}_{init} > 33 \\ \{\mathbf{P}_{\hat{x}_s} : |\mathbf{P}_{\hat{x}_s} - \mathbf{P}_{ref_2}| < \mathbf{P}_{Th_2}\} & : \text{otherwise} \end{cases}, \tag{2.13}$$

where $\mathbf{P}_{ref_1}$ is the median of the three least variances, $\mathbf{P}_{ref_2}$ the median of the least 10% variances, $\mathbf{P}_{Th_1} = 1$ dB and $\mathbf{P}_{Th_2} = 3$ dB. Finally, σ_η^2 is estimated as the average variance calculated from Q, i.e.,

$$\hat{\sigma}_\eta^2 = \frac{1}{|Q|} \sum_{\mathbf{P}_{\hat{x}_s} \in Q} \frac{255^2}{10^{\frac{\mathbf{P}_{\hat{x}_s}}{10}}}. \tag{2.14}$$

The benefits of the proposed adaptation to noise is shown in Fig. 2.3, which shows the average estimation error in the 30 to 40 dB band for the noise-adaptive and non noise-adaptive robust estimation. Adapting the robust estimation threshold and reference to noise decreases the error for less noisy images. The proposed noise estimation algorithm can be summerized using Algorithm 2.

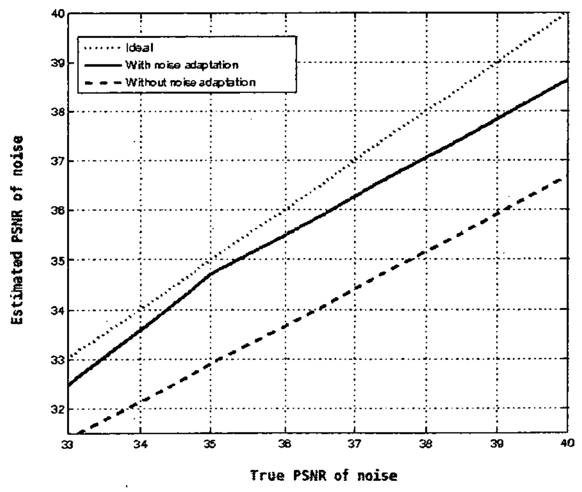


Figure 2.3: Adapting the robust estimation stage to noise reduces errors in the 30-40 dB range. A curve closer to the ideal line is more accurate.

```
1. Select a set of uniformly distributed seeds $S$.
2. Locate a homogeneous location $\hat{\mathbf{x}}$ from each $s \in S$. To do this, repeats
   steps 3-6 for $R$ iterations:
3. Resample $\{(q_{t-1}^{(n)}, \omega_{t-1}^{(n)})\}_{n=1}^{N}$ for iteration $t-1$ to produce $\{(q_{t-1}^{\prime(n)}, \frac{1}{N})\}_{n=1}^{N}$.
4. Dynamically propagate each $q_{t-1}^{\prime(n)}$ by the dynamic model
   $p(\mathbf{x}|\mathbf{x}_{t-1} = q_{t-1}^{\prime(n)})$ and obtain $\{(q_t^{(n)}, \frac{1}{N})\}_{n=1}^{N}$ using (2.6).
5. Weight each $q_t^{(n)}$ by $p(\mathbf{z}_t|\mathbf{x}_t = q_t^{(n)})$ to obtain $\{(q_t^{(n)}, \omega_t^{(n)})\}_{n=1}^{N}$ using (2.9).
6. Find the most homogeneous location from that seed using (2.5).
7. Calculate the set of block variances $\sigma_{B_{\hat{\mathbf{x}}}}^2$ from $\{\hat{\mathbf{x}}_s\}$ using (2.10).
8. Estimate $\sigma_\eta^2$ from $\{\sigma_{B_{\hat{\mathbf{x}}}}^2\}$ robustly using (2.14).
```

Algorithm 2: Proposed noise estimation algorithm.

2.4 Results

2.4.1 Parameters and Limitations

We use five particles (i.e., $N = 5$), 6 iterations (i.e., $R = 6$) and 100 seeds locations (see Fig. 2.2). This leads to a total of 3×10^3 homogeneity measurements. We use $\Sigma_D = [0.1 \times 10^2, 0; 0, 0.1 \times 10^2]$ (see (2.6)) and $\Sigma_S = [0.15 \times 10^2, 0; 0, 0.15 \times 10^2]$ (see (2.11)). We use a block size of $W = 5$ for homogeneity measurement and calculate the block variance with a block size of $W = 7$. The price of eliminating full search while retaining estimation accuracy is the need to define new parameters governing the dynamic movements of blocks such as the number of seeds and the number of iterations R. While the number of seeds appears to be a new parameter, it is not. It relates to the block separation parameters that are commonly defined in other block-based methods (e.g. [2]). On the other hand, the number of iterations $R = 6$ is found to be experimentally sufficient for most blocks to reach homogeneous areas. Robust estimation removes blocks which did not have enough time to reach nearby homogeneity.

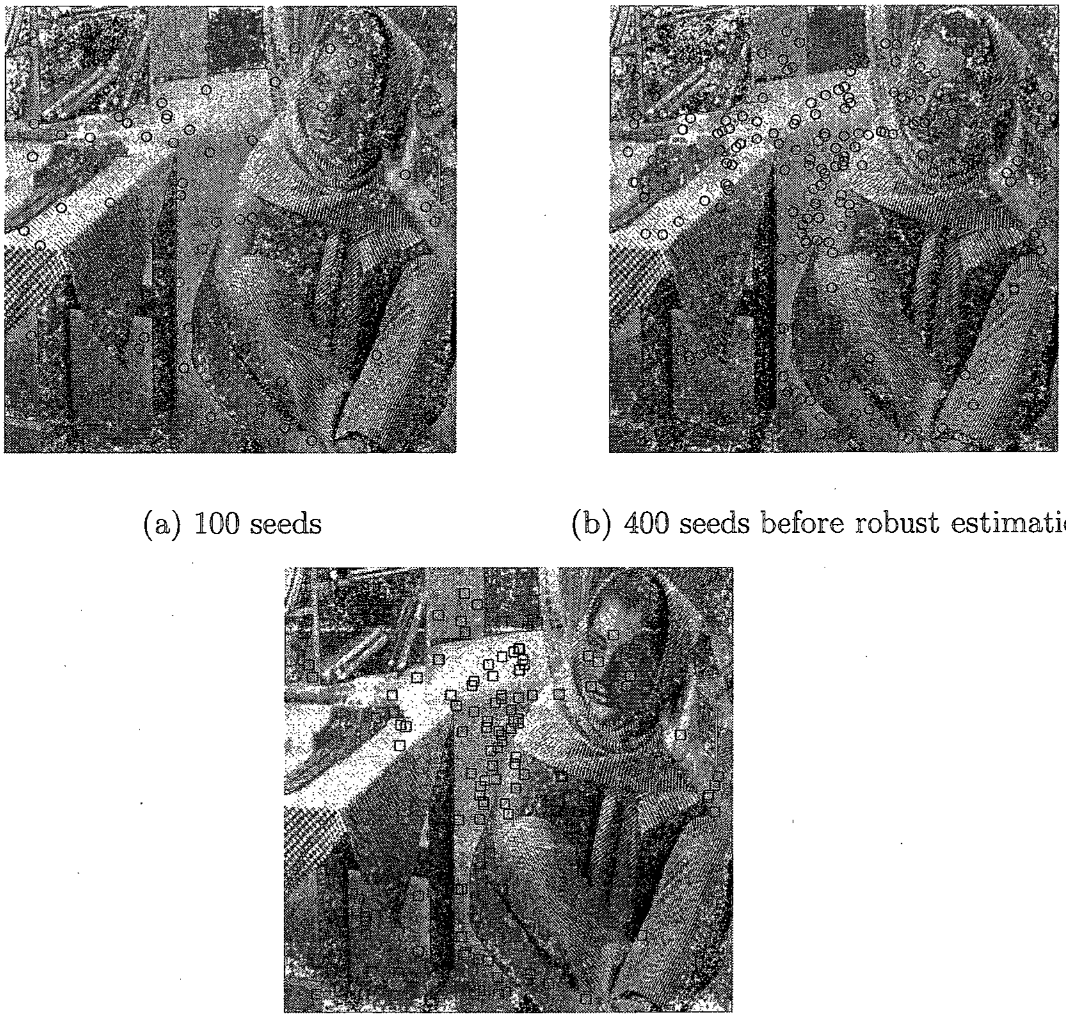
The proposed approach takes advantage of the fact that most images have sufficient homogeneous areas for estimation. If an image is highly structured, not all particles will reach homogeneous areas. The proposed robust estimation reduces this problem, but can be sensitive in the initial estimation threshold area due to the hard threshold.

2.4.2 Objective and Subjective Results

The aim of the proposed algorithm is to retain or exceed the accuracy of block-based methods (e.g., [2]) while eliminating the need for a full search for homogeneity. We compare the proposed method to the wavelet-based method in [3], the robust estimation method in [4], the block-based method with the Laplacian structure detectors in [2], and the block-based method with the variance structure detector in [1]. Block-based methods are faster than wavelet-based or robust estimation based methods because they operate on blocks instead of pixels. The proposed method reduces the complexity of [2] further by increasing the spatial separation between the blocks, while making these blocks seek nearby homogeneity. Other low complexity methods [30,36] tend to overestimate the noise variance in low or high noise. We calculated the number of homogeneity measurements (as a measure of complexity) required by the proposed and the block-based methods [1,2] to process a 512×512 image. The results is summarized in Table 2.1.

Table 2.1: Reduction in the number of homogeneity measurements between proposed and referenced block-based methods. [1] uses a block height of 3 to limit the number of lines to cache in the memory.

Algorithm	# Measurements	Measurement Type	Reduction %	Block size
Proposed	3×10^3	Laplacian	–	5×5
[2]	1.04×10^4	Laplacian	71%	5×5
[1]	6.72×10^3	Variance	55%	13×3



(a) 100 seeds (b) 400 seeds before robust estimation

(c) 400 seeds after robust estimation

Figure 2.4: The effect of increasing the number of seeds from 100 to 400 in 30 dB *Barbara*. More homogeneous areas are detected by increasing the number of seeds, which leads to more homogeneous blocks for the noise estimation stage, but it also significantly increase the number of measurements. Note that for every marked final location in (a) and (b), the algorithm checked 30 blocks (5 particles and 6 iterations) to find it.

To test the performance of the proposed method we use the images in Fig. 2.5 which represent an increasing level of structure. The images are overlaid with AWGN noise ranging from 20 ($\sigma_\eta = 25.5$) to 40 dB ($\sigma_\eta = 2.55$).

First, we examine the effect of increasing the number of seeds from 100 to 400. As seen in Fig. 2.4(b), more homogeneous areas are detected by increasing the number

(a) Cos2 (b) Lena (c) Field (d) Barbara (e) Kiel (f) Trees (g) Baboon

Figure 2.5: Images used for testing.

of seeds, which leads to more homogeneous blocks for the noise estimation. While this also leads to more outliers with the only R iterations. the robust estimator further reduces the outliers before estimation as seen in Fig. 2.4(c). Next, we test the effect of changing the homogeneity measurement types. We test two types, namley: 1) the Laplacian-based with $W = 5$; and 2) the variance-based with $W = 5$ and $W = 9$. We choose a seed in a structured area (see the square in Fig. 2.1) and run the proposed algorithm with the homogeneity measurement types 250 times for 20, 30 and 40 dB *Barbara*. The results are shown in Figs. 2.6 and 2.7. The performance of the Laplacian-based homogeneity measure improves for less noisy images. The variance-based measure performs similarly for all noise levels, but is not as responsive to structure as the Laplacian-based. Increasing the block size from $W = 5$ to $W = 9$, does not seem to affect the performance of the variance-based structure detector as seen in Figs. 2.6(b) and 2.7.

The number of iterations used, i.e., $R = 6$, is not enough in all attempts to move the seeds to homogeneous areas for the Lapacian-based measure for the 20 dB case, however, the majority of particles still move to homogeneity (note how they spread to the left towards homogeneity more than the right towards structure). Increasing the number of iterations to $R = 10$ reduces this problem as seen in Fig. 2.8.

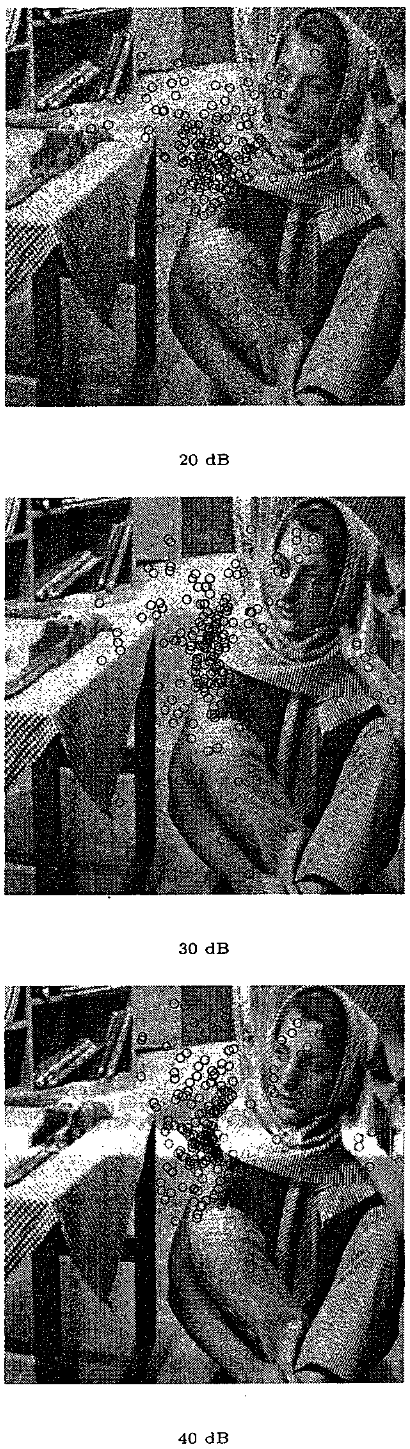
Fig. 2.9 shows the detected homogeneity locations for test images. In all cases, the majority of particles reach homogeneous areas by the end of iterations. Robust estimation then removes the outliers particles from the final estimation process as seen in Fig. 2.10.

The accuracy of the proposed method for noise estimation is measured as the error in dB between the estimated noise peak signal to noise ratio $\hat{\mathbf{P}}_\eta$ and the true value $\mathbf{P}_\eta$. This is depicted in Fig. 2.11 for the proposed and referenced methods and different noise levels ranging from 20 dB to 40 dB (a curve closer to the ideal line indicates more accuracy). The mean and variance of the error are shown in Fig. 2.12. The performance of the proposed method is more consistent than referenced methods.

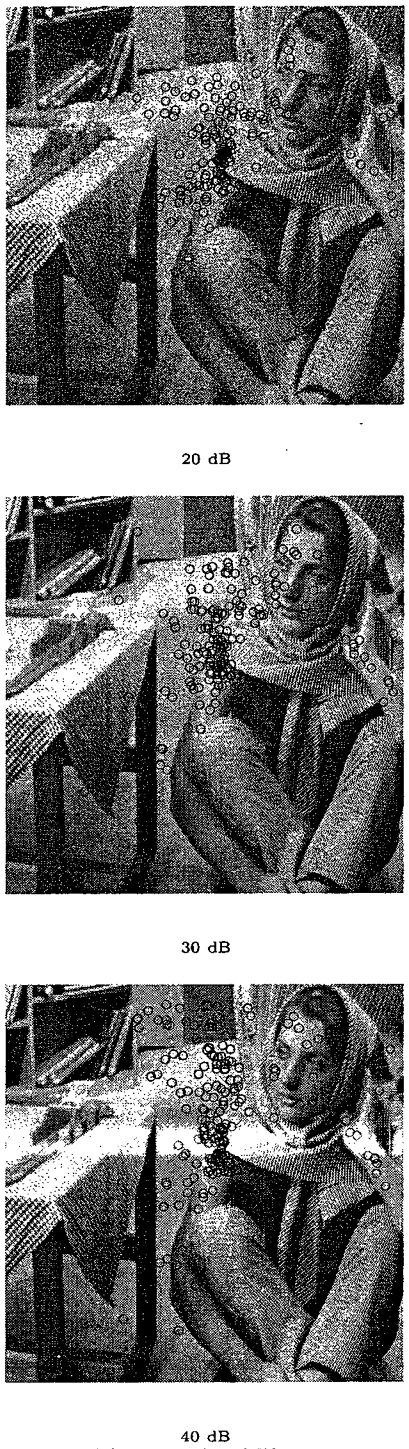
We tested the the proposed method on a JPEG compressed noise-free *Lena* with a q-factor of 40 in Fig. 2.13(b). The proposed method is able to find the homogeneous areas in the compressed image, however, since compression artifacts cannot be modeled as Gaussian, the variance of the blocks in the found homogeneous regions does not correlate with the level of artifacts. When adding noise to a compressed image, the proposed noise estimation method works within expected performance bounds as in Fig. 2.13(a).

2.5 Summary

This chapter proposed a method to use particle filtering for homogeneity localization in noisy images, and applied this method to the problem of estimating the variance of additive white Gaussian noise. The proposed method starts from uniformly distributed spatial seeds and uses particle filtering techniques to guide those seeds to nearby homogeneous locations. The proposed method estimates the AWGN variance using a robust estimator over the block variances in homogeneous locations. The proposed method adapts the robust estimation stage to noise to reduce the estimation error for less noisy images. Our results show that the proposed method reduces the number of homogeneity measurements required by conventional block-based methods while improving the performance. Note that the accuracy level achieved by the proposed method is less than 3 dB making it suitable for the subsequent video processing algorithms (i.e., segmentation) [2].

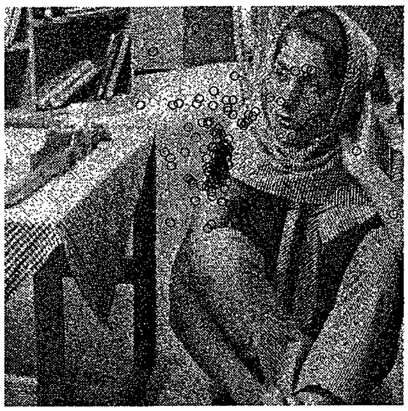


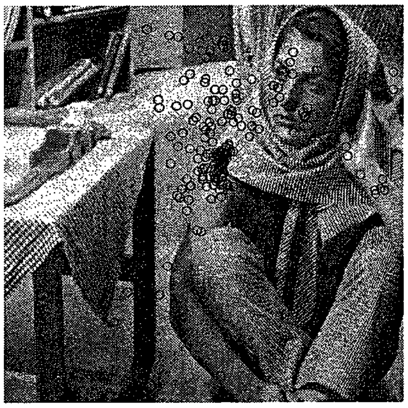
(a) Laplacian-based $W = 5$.



(b) variance-based $W = 5$.

Figure 2.6: Selecting a seed in a structure area (see the square in Fig. 2.1), the result of running the algorithm from this seed 250 times for (a) the Laplacian-based and (b) the variance-based homogeneity detector with $W = 5$ for 20 dB *Barbara* (top row), 30 dB (middle row), and 40 dB (bottom row) . The Laplacian-based is more responsive to structure in less noisy images.


20 dB


30 dB

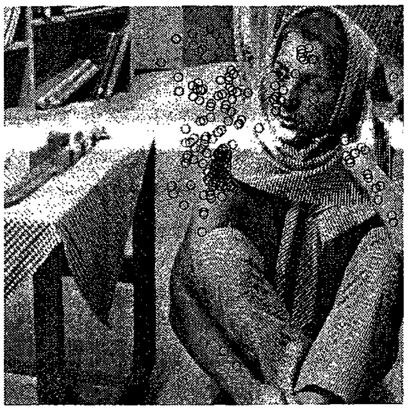
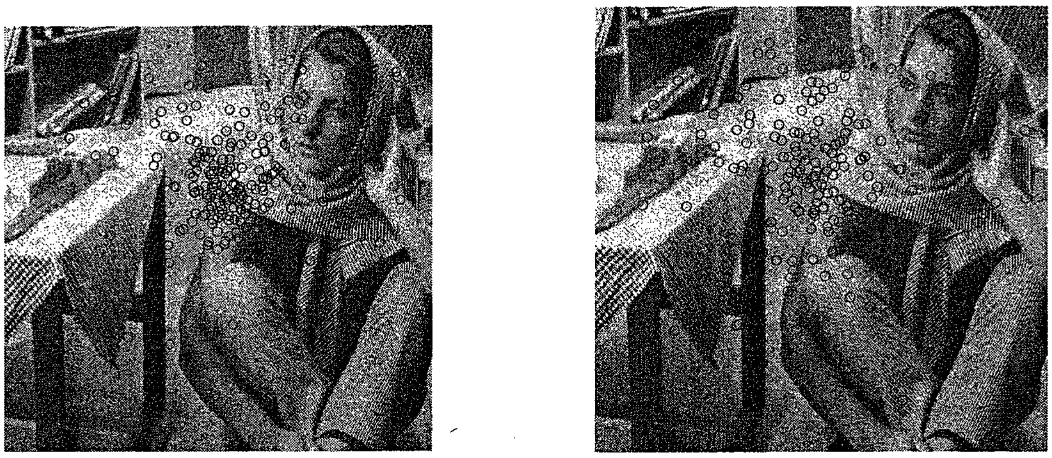
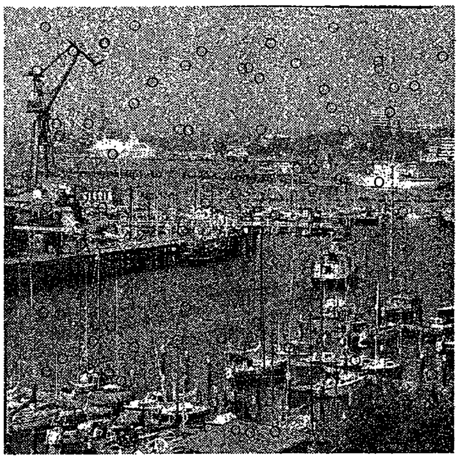

40 dB

Figure 2.7: Selecting a seed in a structure area (see the square in Fig. 2.1), the result of running the algorithm from this seed 250 times for the variance-based homogeneity detector with $W = 9$ for 20, 30, and 40 dB *Barbara*. Increasing the block size from $W = 5$ to $W = 9$ does not improve the performance of the variance-based detector.

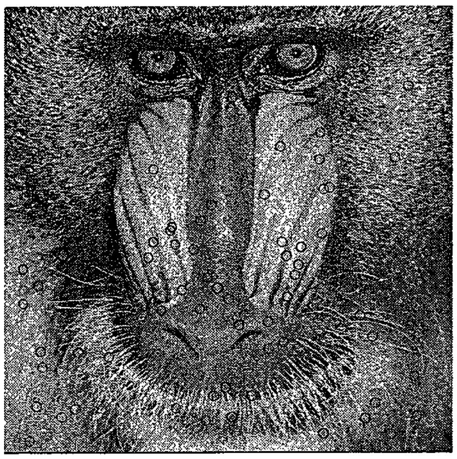


(a) $R = 6$ (b) $R = 10$

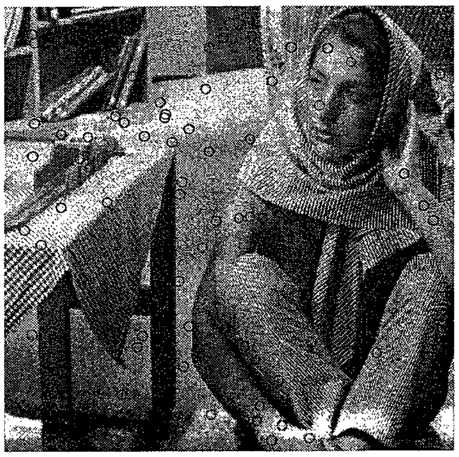
Figure 2.8: In a 20 dB *Barbara* image, increasing the number of iterations from $R = 6$ to 10, allows the Laplacian-based structure detector to move most of the seeds towards homogeneity.



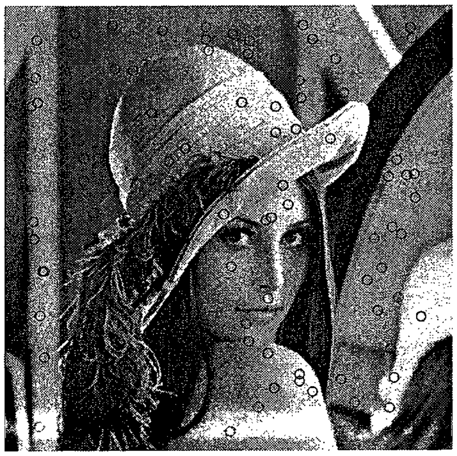
(a) Kiel 20 dB



(b) Baboon 25 dB

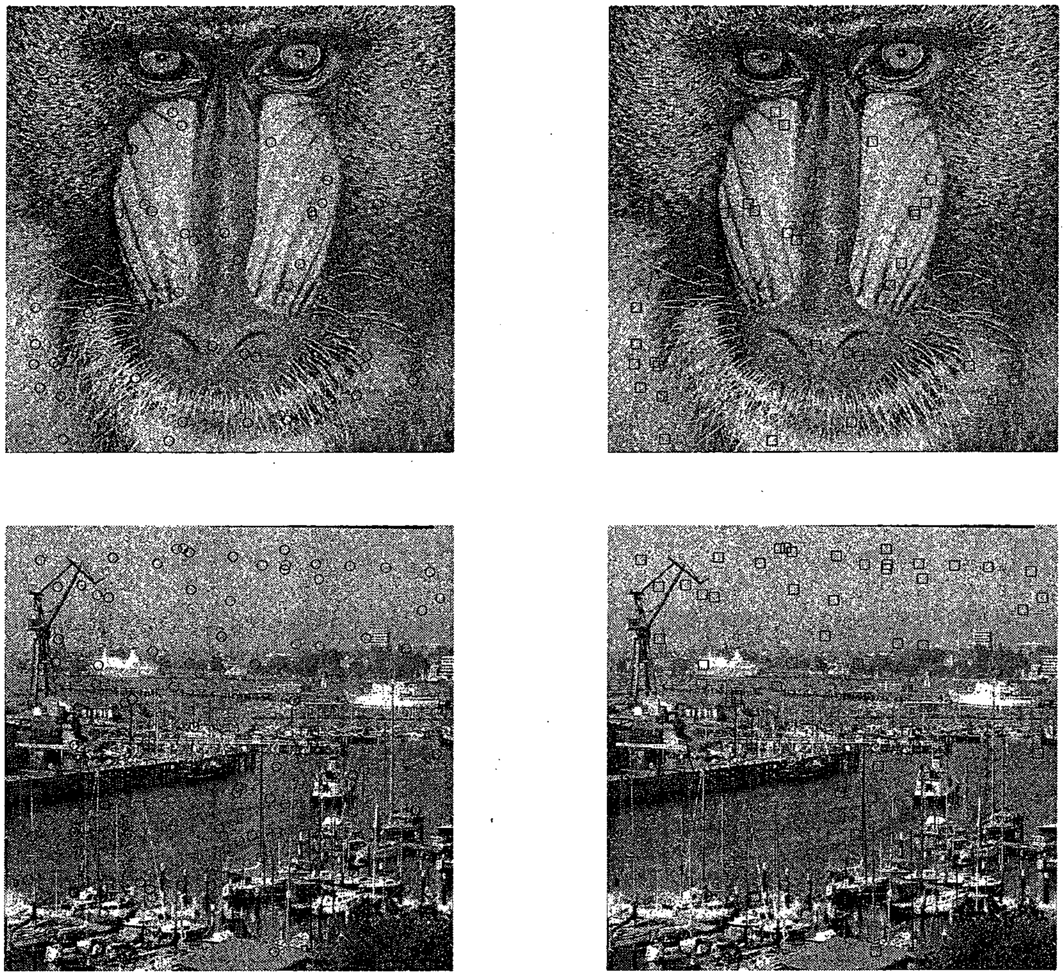


(c) Barbara 30 dB



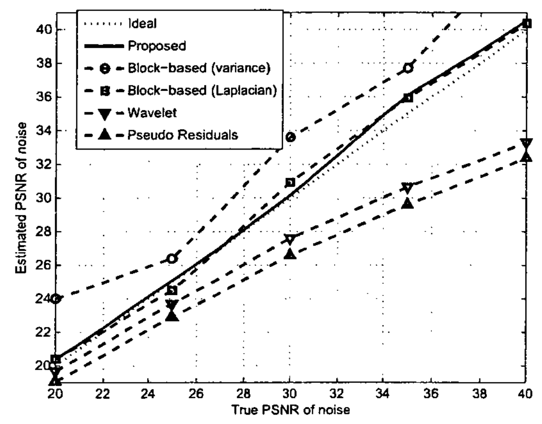
(d) Lena 40 dB

Figure 2.9: The detected homogeneous locations by the particle filters (before robust estimation) from test images at different noise levels. In all cases, a sufficient number of particles reach homogeneous areas. Note that for visualization purposes, the circles used are 11 pixels in diameter. The actual blocks are squares of size 5×5 at the center of the circles. The initial seeds used for all images are the same as the ones shown in Fig. 2.2.

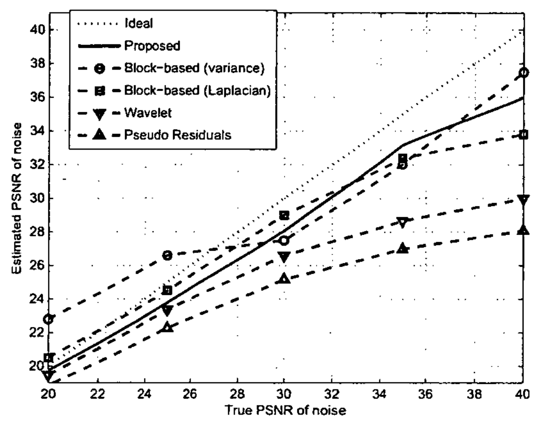


(a) Detected homogeneous locations (b) Locations used in noise estimation

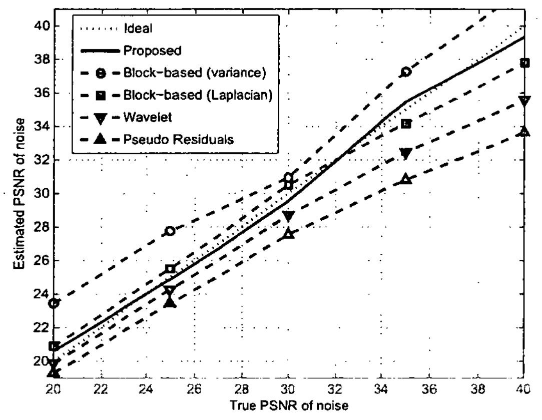
Figure 2.10: For 20 dB *Baboon* and 25 dB *Kiel*, (a) the locations, marked with circles, detected to be homogeneous by particle filtering, and (b) the locations, marked with squares, used in noise estimation after the robust estimator removes outliers.



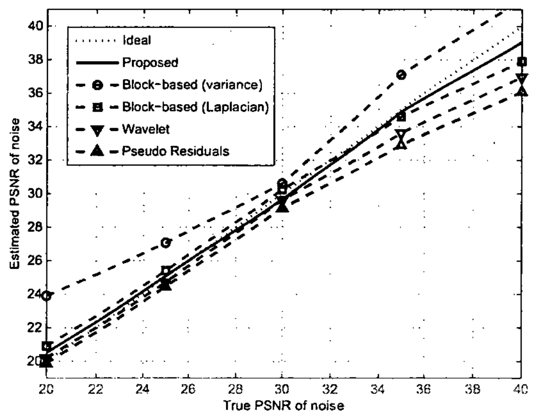
(a) Kiel



(b) Baboon

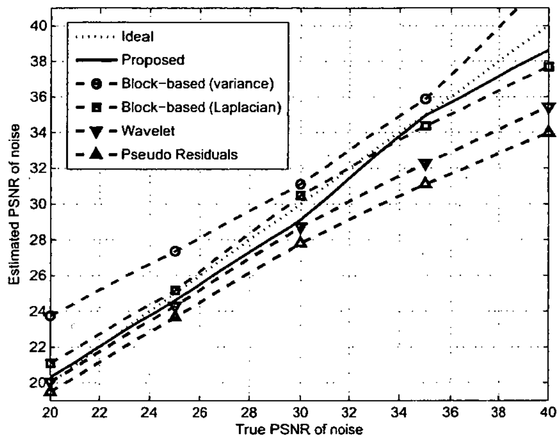


(c) Barbara

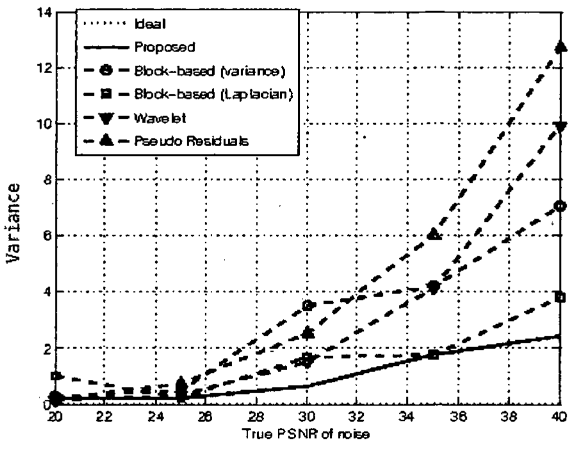


(d) Lena

Figure 2.11: Accuracy of the proposed method compared to the block-based method in [1] with variance homogeneity measurement, the block-based method in [2] with Laplacian-based homogeneity measurement, the wavelet-based method in [3], and the pseudo residuals followed by robust estimation method in [4]. A curve closer to the ideal line is more accurate.

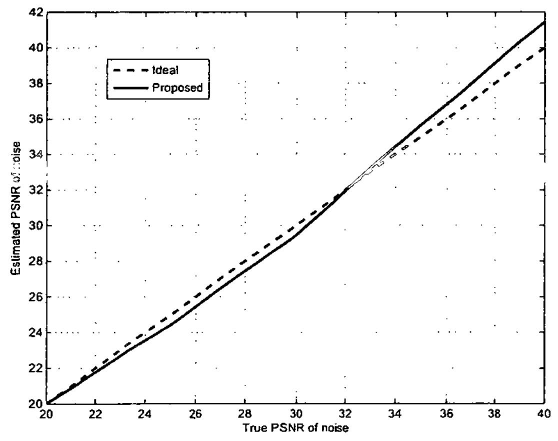


(a) Mean

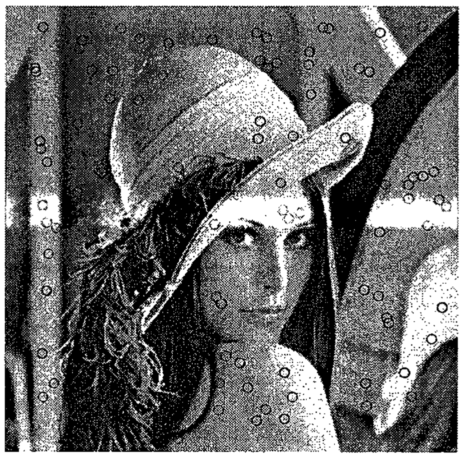


(b) Variance

Figure 2.12: The mean and variance of the estimation error over all test images in Fig. 2.5 and noise levels (i.e., 20-40 dB). The proposed method is consistently more accurate than the referenced methods on used test images. A curve closer to the ideal line is more accurate.



(a) Lena (q-factor=40)



(b) Located homogeneous areas

Figure 2.13: Testing on compressed images: (a) noise estimation result from adding noise to compressed Lena image, (b) located homogeneous areas on a compressed Lena image.

Chapter 3

Object Segmentation with Fixed and Moving Cameras

3.1 Introduction

Segmentation, in this thesis, is the process of classifying pixels to foreground or background based on motion. The classification accuracy, temporal stability, and computational complexity are the key performance measures. In this chapter, we propose a fast segmentation approach based on the integration of a gray-level background subtraction technique with a mixture of Gaussians background maintenance approach. The fast segmentation approach retains on average 96% of the accuracy of the background subtraction technique, while eliminating its reliance on a fixed background model. This chapter also proposes a segmentation approach for video sequences with global motion. The approach starts by estimating and compensating the global motion of neighboring frames to create multiple observations of the current frame with the objects of interests moving to reveal background areas. The observations are used to estimate the parameters of a mixture of Gaussians model and extract a background frame using less computation than a recent median-based technique. We further reduce the complexity of the recent technique by alternating between full background model estimation using multiple motion-compensated neighbours, and single motion compensation of the previous frame's generated background model. Generated background frames are used by a color based background subtraction technique followed by global thresholding and morphological post-processing to obtain the final segmentation output. The proposed method is several times faster than a recent technique, while retaining or exceeding its accuracy.

3.2 Review of Related Work

Many video object segmentation methods are presented over the years. The type of video object segmentation used is dictated by the application requirements. Some of the major requirements affecting which method to use are: 1) real-time operation; 2) adaptation to environmental conditions; and 3) accuracy. For example, fast video surveillance systems must run in real-time and account for illumination changes in case of outdoor surveillance.

We can divide object segmentation techniques into two categories [37]: motion segmentation and change detection. Motion segmentation classifies motion vectors into groups that fit particular motion models. This leads to one object with articulated motion to become segmented into multiple parts (each fitting a model). In some applications, e.g, articulated object tracking, this is a desired result, while in other applications, e.g., rigid object tracking, it further complicates the tracking problem.

3.2.1 Segmentation with fixed camera

Segmentation by change detection methods have the different objective of labeling pixels in the frame as changed or unchanged. This is usually done in two stages. The first stage is background subtraction [38–43], which is followed by thresholding

[44–50]. The performance of change detection methods is reliant on the accuracy of the background model. For example, for indoor surveillance with a fixed camera and lighting conditions, background subtraction followed by thresholding produces good results. With outdoor surveillance, the background model changes over time and requires adaption for accurate results.

Many algorithms are presented over the years to adapt the background model. These algorithms can be classified according to [51] into temporal median [52], single Gaussian [53], mixture of Gaussians [54–58], and kernel density estimation [59, 60]. The main idea is to find pixels deviating from an updated background model. In the simplest case of a temporal median filter [52], foreground pixels are considered outliers rejected by the median operation. However, the size of the median filter and the speed of the moving objects affect the performance. There is also the added computational cost of performing the median operation on all pixels in the frame. In the single Gaussian case [53], the background pixel is assumed to temporally follow a Gaussian distribution. If an observed pixel value significantly deviates from this model, it is considered a foreground pixel and is rejected. The single Gaussian method does not take into account dynamic backgrounds such as ones resulting from water waves or the effect of wind on trees. This motivates the use of the mixture (or multiple) of Gaussians model [54–58]. In this model, the background is represented by more than one Gaussian. Note that the Gaussian and mixture of Gaussians approaches require a period of training to estimate the model parameters. Kernel density estimation techniques [59, 60] are nonparametric as they estimate the density function from observed pixels regardless of their actual distribution. Background adaptation or modeling techniques [53–60] are followed by thresholding of the probability density function to label pixels as changed or unchanged. In other words, they are segmentation methods in their own right.

In summary, change detection followed by thresholding yields connected blobs that are suitable for object tracking provided the background or reference frame is fixed. This is not the case with outdoor surveillance. The statistical mixture of Gaussians based background modeling methods are better suited for outdoor surveillance, because they handle dynamic backgrounds. However, the blobs produced often appear fragmented due to similarity between the colors of the objects and backgrounds leading to wrong classification of foreground pixels as background. We aim at integrating the two methods to produce a segmentation method that results in suitable blobs for tracking under varying illumination conditions.

3.2.2 Segmentation with global motion

Change detection is a more challenging problem if the camera is moving. The case of a moving camera is identical to that of static camera if motion is estimated and compensated for [61], e.g., using [62, 63]. The method in [63] is more oriented to video object segmentation since there is a feedback loop from segmentation to motion estimation for the mutual benefit of both algorithms. The segmentation method used in [63] identifies local motion outliers and the global motion estimation and compensation method reduces frame differences due to global motion. Note that, in case of a moving camera, frame subtraction is used instead of background subtraction. Frame subtraction suffers from the aperture problem, i.e., when a homogeneous area appears to be static when it is in fact moving. The effect of the aperture problem is reduced using spatial techniques such as in [64], nevertheless, it remains of major influence on the performance.

A class of methods [6, 65, 66] use sprite generation to create a better model of the background. The main idea of [6] is to estimate the global motion between a number of neighboring frames and the current frame and then warp or project these frames into the current frame. During the projection and using median filtering, the foreground object is removed (to a degree) from the current frame effectively producing a background frame. Background subtraction and thresholding are then used to produce the segmentation results. These algorithms have found promising application in video coding. It can also be helpful in video surveillance processing, however, the computational cost for fast video surveillance is significantly high due to the multiple global motion estimations for every frame. Another approach is to impose restriction on the type of camera motion to handle and accept some loss of segmentation accuracy to gain real-time performance. The method in [67] is an example of such approach. The authors of [67] estimate and compensate for global motion using feature points in the current and reference frames.

The outline of the chapter is as follows. Sections 3.3 and 3.4 present the proposed segmentation methods for fixed and moving cameras, respectively. Section 3.5 presents the results, and Section 3.6 summarizes the chapter.

3.3 Proposed Segmentation with Fixed Camera

For fast segmentation with no global motion, we integrate the background subtraction based approach in [68] and the background maintenance approach in [5] as summarized in Fig. 3.1. This integration starts by generating a difference frame D_l between the current frame F_l and a background model G_l, i.e., $D_l = |F_l - G_l|$. We use the method in [5] to generate G_l. The details of background generation are discussed in Section 3.4 presenting the segmentation under global motion method. The difference between background generation in the fixed and moving camera case are: 1) the means in the background update approach for fixed cameras are initialized to 128, whereas the means for the background update for moving cameras are initialized to the pixels in the current frame for faster convergence to background pixels; 2) the hysteresis thresholds for the fixed camera case are higher than in the moving camera case since the frames are perfectly aligned due the lack of motion; and 3) when compensating for motion, we may get black regions around object boundaries which are rejected in the moving camera case. D_l may be degraded by three main factors; noise, motion instability, and local changes such as shadow and object velocity. Noise introduces globally scattered artifacts in D_l whereas both motion instability and local changes lead to holes and gaps in the detected moving regions. We reduce noise and limit the gray-levels (i.e., motion difference) in D_l.

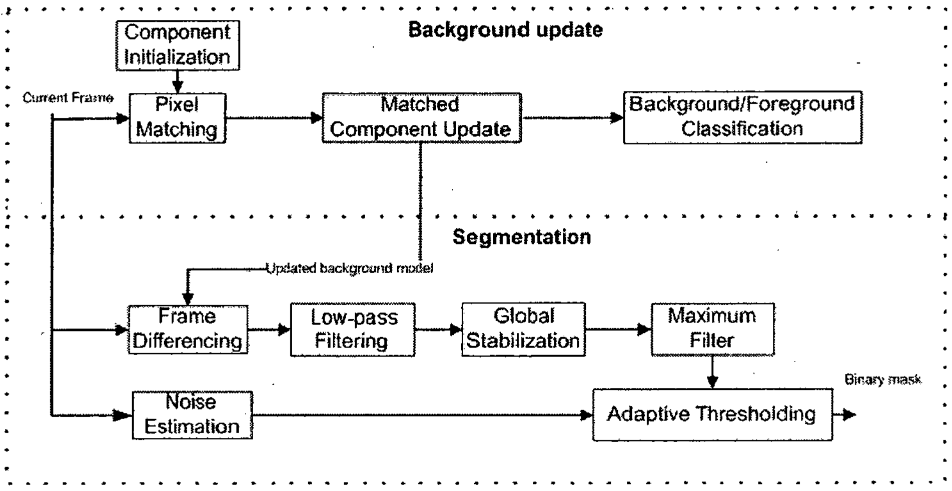


Figure 3.1: Block diagram of the segmentation method with fixed camera.

First, D_l is low-pass filtered to reduce the effect of noise using

$$D_l^{\mu} = LP(F_l - G_l), \tag{3.1}$$

where $LP()$ is a 5×5 low-pass (i.e., moving average) filter. Then, we limit the differences in the noise-reduced difference frame D_l^{μ} (μ here denotes the result from finding the mean value of neighbors within a block surrounding the current pixel) to a gray-level limit g_{lim}. This step is necessary to globally stabilize (or filter) the high motion in D_l. High motion may be caused by 1) local changes and 2) object textures and velocities. Setting g_{lim} low (e.g., 31) increases the sensitivity to artifact caused by non-motion related factors (such as shadow) and produces larger regions, but is good for low illumination or low contrast video sequences. A high g_{lim} (e.g., 63) is good for frames with low noise-levels as it achieves more accurate object boundaries. It is important to establish a lower-bound on g_{lim}. This is because a too low g_{lim} will cause many pixels in D_l to be wrongly classified as white (moving) pixels in the binary frame B_l. To establish such a lower-bound, let P_η be the Peak Signal to Noise Ratio (PSNR) defined by $\text{P}_\eta = 10\log_{10}\frac{g_{max}^2}{\sigma_\eta^2}$, where g_{max} is the maximum possible gray-level (e.g., 255) and σ_η^2 is the variance of the additive white Gaussian noise. We condition $\text{P}_{D^\mu}^{g_{lim}}$ (the PSNR of D_l^μ obtained with a given g_{lim}) to be larger than P_η or

$$\text{P}_{D^\mu}^{g_{lim}} = 10\log_{10}\frac{g_{lim}^2}{\text{MSE}_{D_l^\mu}} \geq \text{P}_\eta, \tag{3.2}$$

where $\text{MSE}_{D_l^\mu}$ is the mean-square error (MSE) between noisy D_l and its noise-reduced approximation D_l^μ. When a 3×3 averaging filter is applied to get D_l^μ, it can be shown that $\text{MSE}_{D_l^\mu} = \frac{1}{72}\sigma_\eta$ [69]. Let $g_{lim} = \alpha g_{max}$, (3.2) is rewritten as

$$\text{P}_{D^\mu}^{g_{lim}} = 10\log_{10}\frac{g_{max}^2}{\sigma_\eta^2} + 10\log_{10}72\alpha^2 \geq \text{P}_\eta. \tag{3.3}$$

From (3.3), we get $\alpha \geq 0.1179$ and $g_{lim}^{min} \geq 30$. From (3.2), we note that $\text{P}_{D^\mu}^{g_{lim}}$ is calculated from g_{lim} instead of $g_{max} = 255$. This is because after globally stabilizing D_l, all gray-levels will be limited to g_{lim}. Since $g_{lim} < g_{max}$, $\text{P}_{D^\mu}^{g_{lim}}$ will be lower than $\text{P}_{D^\mu}^{g_{max}}$. Obtaining B_l from D_l is sensitive to global or local changes, therefore it may fail for low quality videos. To get accurate B_l, the $\text{P}_{D^\mu}^{g_{lim}}$ should be as high as possible.

After stabilizing the high motion in difference frames, a maximum filter is applied to reduce small holes, gaps, and granular blobs inside the moving regions and cause stability around the boundaries between the moving and the static regions. D_l is then adaptively thresholded. First, a spatial threshold t_l^s is obtained using a robust thresholding method [70], which is adapted to noise to obtain t_l^η according to

$$\text{t}_l^\eta = \text{t}_l^s + \gamma\sigma_\eta^2, \quad \gamma < 1. \tag{3.4}$$

t_l^η is quantized to one of three levels to obtain t_l^q in order to stabilize thresholding. This is done using

$$\mathrm{t}_l^q = \begin{cases} \mathrm{t}_l^{min} : & \mathrm{t}_l^\eta \le \mathrm{t}_l^{min} \\ \mathrm{t}_l^{mid} : & \mathrm{t}_l^{min} < \mathrm{t}_l^\eta \le \mathrm{t}_l^{mid} \\ \mathrm{t}_l^{max} : & \text{otherwise} \end{cases} . \tag{3.5}$$

to obtain the final frame threshold t_l used to produce the binary frame B_l, t_l^q is adapted to temporal changes using

$$\mathrm{t}_l = \begin{cases} \mathrm{t}_l^{min} : & \mathrm{t}_l^q \le \mathrm{t}_l^{min} \\ \mathrm{t}_{l-1} : & \mathrm{t}_l^q < \mathrm{t}_{l-1} \\ \mathrm{t}_l^q : & \text{otherwise} \end{cases} . \tag{3.6}$$

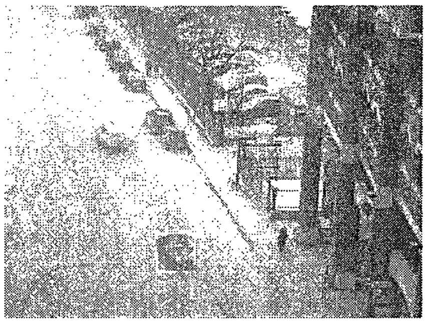
More details on this thresholding schemes are in [68].

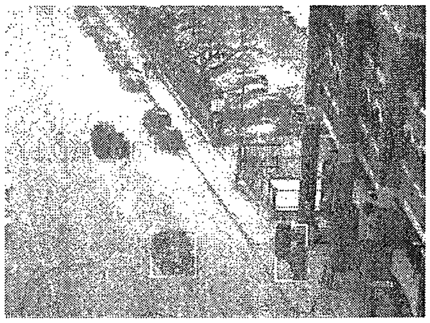
Next, we apply morphological edge detection and contour tracing to the binary output of the previous step. We use the object contour chain codes to perform contour filling. Contour filling allows us to traverse object pixels from left to right and from top to bottom to extract key features such as dominant color and area. It can also be used to generate a color histogram of the object pixels without the background outliers associated with taking the histogram of the object's minimum bounding box. Let Q^o be the sequence of spatial points in the object contour with coordinates (Q_x^o, Q_y^o). For every Q_x^o for a given Q_y^o, we keep a label indicating if the contour's chain code as traversed clockwise is increasing, decreasing or zero as in

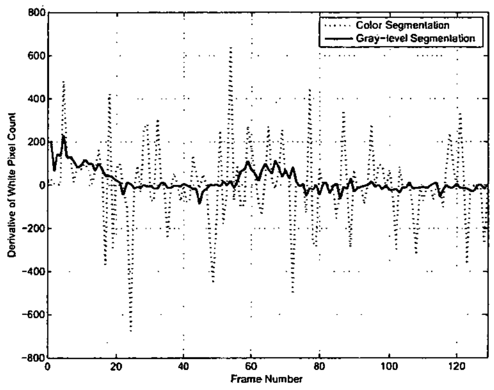
$$L_{Q_x^o} = \mathrm{sgn}(\mathrm{sgn}(Q_x^o - Q_x^{o-1}) + \mathrm{sgn}(Q_x^{o+1} - Q_x^o)). \tag{3.7}$$

We then use every Q_x^o for a given Q_y^o and its label $L_{Q_x^o}$ and keep filling until another point is reached with the complement label $-L_{Q_x^o}$. After contour filling, we obtain a list of image objects in the current frame. This list is used as one of the inputs to tracking in Chapter 5. We use gray-level background subtraction instead of color based background subtraction (which we later use for segmentation with global motion) because it yields connected blobs, due to the use of the maximum filter, eliminating the need for morphological post-processing. Moreover, generated blobs are more temporally stable due to the quantization of binary thresholds. While the segmentation output of the color-based background subtraction is more accurate than that of the gray-level segmentation around object boundaries, the temporally stable connected blobs produced by the gray-level approach are more advantageous to tracking. This is illustrated in Fig. 3.2 which shows the tracking results for an online outdoors sequence with the color and gray-level segmentation techniques. While the track is lost due to temporal instability for the color segmentation case, the track is maintained for the gray-level case. This could be attributed to the lack of contrast in the video. We measure temporal stability using the derivative of white pixel count (i.e., object area) over time. We can see that the gray-level segmentation is more stable from the plot of the derivative than the color-based segmentation.

To show the importance of accurate noise estimation in adapting segmentation to noise, we show the visual output and F-measure (i.e., objective measure using ground truth defined later) comparison for 25 dB noisy *Hall Monitor* sequence at different levels of noise estimation errors in Fig. 3.3. As the estimation error increases, the segmentation accuracy decreases. The segmentation accuracy decreases for even 1 dB error. This goes to show the importance of accurate noise estimation to segmentation.


(a) Color Segmentation


(b) Gray-level Segmentation

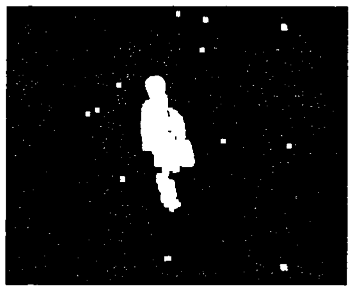


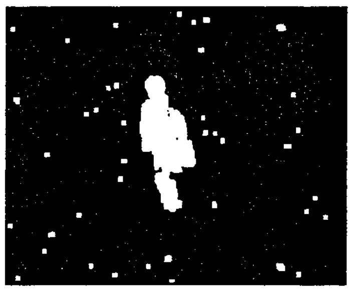
(c) Objective Measurement of Temporal Stability

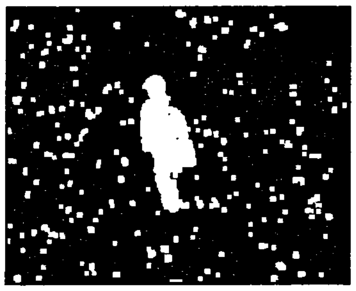
Figure 3.2: Temporal instability in segmentation causes track loss in the case of color segmentation in low contrast compared to the gray-level segmentation case. Temporal stability is measured using the derivative of object area (white pixel count) in (c) and shows how the gray-level segmentation is more temporally stable.

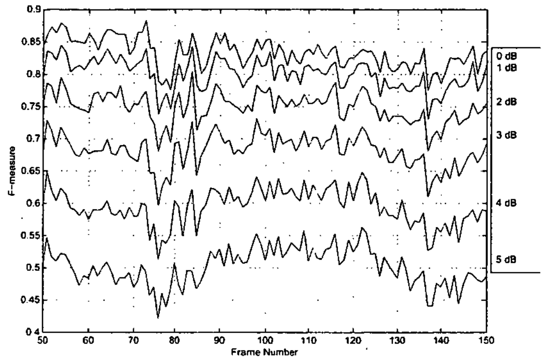
3.4 Proposed Segmentation with Global Motion

We divide the process of segmentation with global motion into the three stages in Fig. 3.4. In the first stage, the current frame F_l and its neighboring frames are used to generate a background frame G_l. G_l is then used with F_l in the second stage of segmentation to generate an initial binary frame, which is post processed to generate the final binary frame B_l showing extracted objects. Our approach is based on [6]. Our background generation differs from [6] in two areas: 1) we use a mixture of Gaussians technique to generate the background model instead of their median based technique; and 2) we alternate between a full background model estimation using multiple neighboring frames and motion compensation of the previous frame


(a) 0 dB estimation error


(b) 2 dB estimation error


(c) 3 dB estimation error



(e) F-measure at different levels of noise estimation error

Figure 3.3: Sensitivity of segmentation to noise estimation error: (a), (b), and (c) show visual output at different levels of noise estimation errors for 25 dB noisy *Hall Monitor* sequence; (e) the F-measure comparison using ground truth segmentation. Accurate noise estimation is important to segmentation.

background model as illustrated in Fig. 3.5. Our segmentation differs from [6] in the use of color compensation in generating difference frames and how the adaptive global thresholds are computed.

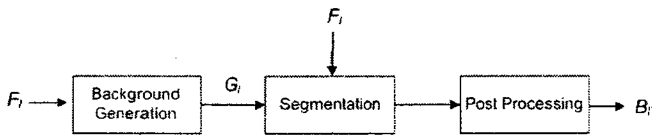


Figure 3.4: Block diagram showing the three stages of the proposed segmentation with global motion method.

In the first stage of the proposed method, i.e., the background generation stage, we use a modified mixture of Gaussians technique to the one in [5] to generate a background model of the frame. We later use the generated background model to perform

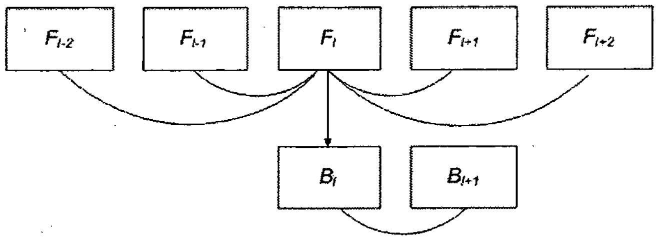


Figure 3.5: In the background generation stage of the proposed segmentation with global motion method, we alternate between full background model estimation using multiple neighboring frames and motion compensation of the previous frame background model.

color-based background subtraction followed by adaptive thresholding to generate the binary frame. The mixture of Gaussians technique assumes a stationary camera and cannot be used as is with moving cameras. We use a video object segmentation based global motion estimation technique to estimate the affine motion parameters describing the camera motion. Then, for each frame F_l, we compensate its neighboring frames F_{l+r}, where $r \in \{-N_r, ..., -2, -1, 1, 2, ..., N_r\}$ and N_r is the size of the neighborhood, using F_l as the reference. In other words, we warp neighboring frames to the coordinate system of the current frame (e.g., as in Fig. 3.6). Thus, we obtain estimated observations of the current frame taken at different times during which the objects of interest are moving. We use these observations to estimate the parameters of the mixture of Gaussians background model G_l, which is the output of this background generation stage detailed in Fig. 3.7. We measure the quality of G_l by comparing it to F_l using the approach in [6]. As we add neighboring frames, we monitor the derivative of the quality measure to stop the process when adding more frames no longer improves the quality of G_l. In [6], the median is used to generate the background model from compensated neighboring frames F_{lr}.

For global motion estimation (GME), we use a method largely based on [63] to estimate the affine global motion model parameters. We differ from [63] in propagating the short term motion parameters to obtain the long term ones. The purpose

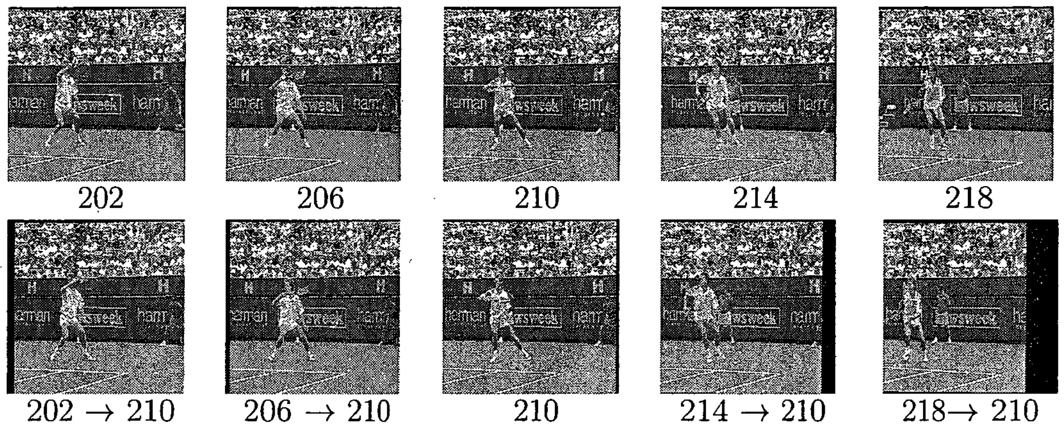


Figure 3.6: Top Row: frames neighboring frame 210. Bottom row: neighboring frames aligned to the current frame through motion compensation. Note how the bottom row represents different observations of frame 210 with the object (the tennis player) moving and revealing the background behind him (i.e., the referee).

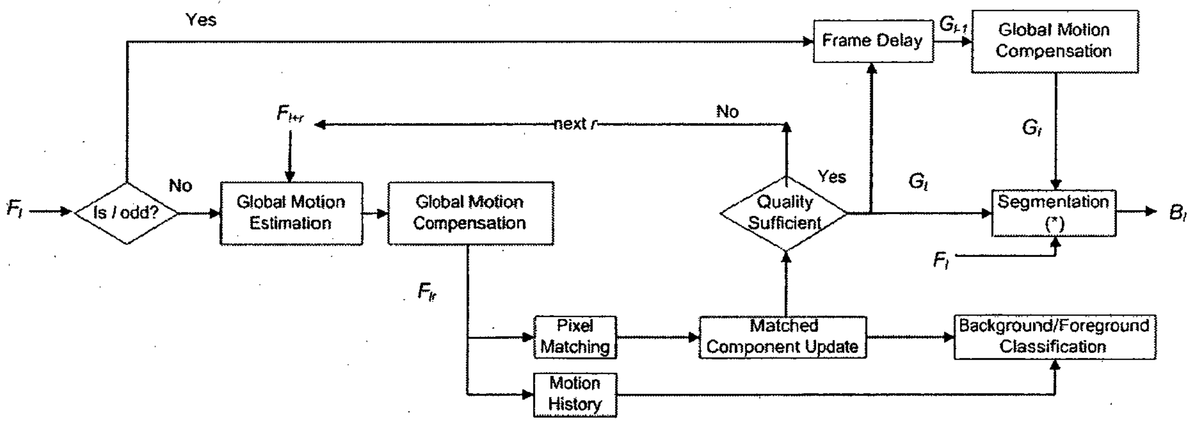


Figure 3.7: Block diagram of the background generation method. The segmentation block marked with (*) is later detailed in Fig. 3.9

of a motion model is to describe the real motion between consecutive frames F_l and F_{l-1} of a video sequence at time instances l and $l-1$. Based on this model, motion parameters are estimated. In GME, a single model applies to the whole frame (compared to block-based local motion estimation). There are different parametric motion models used to describe and estimate GM such as the affine, the projective and the bilinear models. Depending on the selected model, we can control the level of detail of the estimated motion. The 6-parameter affine model is described by

$$\begin{aligned} dx_i &= a_1 + a_2 x_i + a_3 y_i, \\ dy_i &= a_4 + a_5 x_i + a_6 y_i, \end{aligned} \tag{3.8}$$

where (x_i, y_i) is the location of the i^{th} pixel in the current frame F_l, (dx_i, dy_i) is the motion vector of the corresponding pixel from the previous frame F_{l-1} to F_l, and a_1, a_2, a_3, a_4, a_5, and a_6 are the affine global motion parameters, which we group using vector $\mathbf{a}$. We use the affine model in all simulations because it can describe the projected 2D motion of most camera motions.

With the motion model defined, we incorporate it into the Displaced Frame Difference (DFD) estimation criterion, which is based on the *constant-intensity* assumption. This assumption states that the intensity remains constant along motion trajectories. The error E_{DFD} based on the DFD is defined as

$$E_{DFD}(\mathbf{a}) = \sum_{i=1}^{N} \left| F_{l-1}(x_i + dx_i, y_i + dy_i) - F_l(x_i, y_i) \right|^s, \tag{3.9}$$

where N is the total number of pixels in F_l and $s = 1$ in case of the Sum of Absolute Difference (SAD) or $s = 2$ in case of the Sum of Square Difference (SSD). To search for the best set of parameters to describe the global motion, we minimize $E_{DFD}(\mathbf{a})$, or simply $E(\mathbf{a})$, using the Gauss-Newton optimization method. For increased accuracy and improved computational cost, a multi-resolution representation, or hierarchical representation, of an image or video frame, is used. Using this representation, the original frame is rebuilt like a pyramid where the finest level is the original frame. Then, the resolution between successive levels is reduced by half. After the frame pyramid is built, the estimation of the parameters $\mathbf{a}$ starts at the coarsest level and progresses to the next finer level until it reaches the finest level. The result from the previous coarser level is projected to the next finer level as an initial solution or search point. We use the method in [63] for outliers rejection. Using an initial estimate for the translation parameters, we reduce the number of iterations required by the Gauss-Newton optimization method to reach a local minimum. Normally, we use 3-step search for initial estimation in the first six frames and motion parameters prediction from the seventh frame onward. Motion parameters prediction is faster than the 3-step search. However, this scheme of 3-step search followed by prediction is usable when performing GME on consecutive frames. When direct estimation is used to bring non-immediate neighboring frames to the coordinate system of the current frame, a larger search radius is needed in the initial estimation and motion parameters prediction cannot be used. To still benefit from the computational savings of the motion parameters prediction and avoid re-estimation of GM parameters, we propagate the motion parameters for far (i.e., non-immediate) neighboring frames, also known as long-term parameters, using the estimated consecutive parameters, known as short-term parameters, until we reach the current frame. We demonstrate this in Fig. 3.8. The long term parameters for the example in Fig. 3.8 are propagated using

$$\begin{aligned}
a_1^{l-2\to l} &= a_1^l + a_1^{l-1}a_2^l + a_3^l a_4^{l-1};\\
a_2^{l-2\to l} &= a_2^l a_2^{l-1} + a_3^l a_5^{l-1};\\
a_3^{l-2\to l} &= a_2^l a_2^{l-1} + a_3^l a_6^{l-1};\\
a_4^{l-2\to l} &= a_4^l + a_1^{l-1}a_5^l + a_4^{l-1}a_6^l;\\
a_5^{l-2\to l} &= a_2^{l-1}a_5^l + a_5^{l-1}a_6^l;\\
a_6^{l-2\to l} &= a_2^{l-1}a_5^l + a_6^l a_6^{l-1}.
\end{aligned} \tag{3.10}$$

After $2N_r + 1$ observations of the current frame become available, we use them to estimate the parameters of the mixture of Gaussians model. We represent each

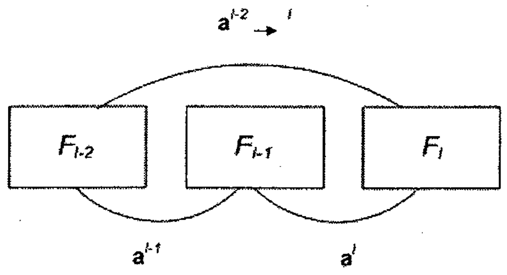


Figure 3.8: Long-term motion parameters are propagated from short-term motion parameters.

pixel in F_l as a random vector Z of three color components (e.g., RGB). Consider r to be the index of the sequence of observations of the current frame F_{lr} obtained by compensating the motions in neighboring frames. Any pixel in F_{lr} belongs to a class $k \in \{1, 2, ..., K\}$, where K is the number of different classes we wish to consider. We use $K = 3$.

We estimate which class k lead to the current pixel observation $z \subset Z$. This is now a pattern classification problem of obtaining the maximum posterior probability $P_r(k|z)$ governed by Bayes's theorem

$$P_r(k|z) = \frac{P_r(z|k)P_r(k)}{P_r(z)}, \tag{3.11}$$

where $P_r(z|k)$ is the likelihood probability of z corresponding to k, $P_r(k)$ is the prior probability of a class k and $P_r(z)$ is a scaling factor. We use a Gaussian model for the distribution of Z given a class k

$$P_r(z|k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_{k,r}|^{\frac{1}{2}}} e^{-0.5(z-\mu_{k,r})^{\mathbf{T}}\Sigma_{k,r}^{-1}(z-\mu_{k,r})}, \tag{3.12}$$

where $\mu_{k,r}$ and $\Sigma_{k,r}$ are the mean and covariance of $P_r(z|k)$ at r, D is the dimension of Z, and $\mathbf{T}$ is the transpose. $P_r(k)$ is a weighting related to the frequency of a certain color k. We assume $\Sigma_{k,r} = \sigma_k^2 I$, where I is the identity matrix, to avoid the expensive matrix inversion at the expense of some loss in accuracy. With this formation, the distribution of the next pixel Z in the observation F_{lr} is characterized by a mixture of Gaussians model. To form the class probabilities $P_r(k)$, we use a weighting factor ω_k which records how many pixels where matched to class k. As we get new pixel values, we classify them into a given class k and use their value to update our model. The classification is done using

$$max\left[|F_{lr}^{Red}(x_i,y_i)-\mu_{k,r}^{Red}|,|F_{lr}^{Green}(x_i,y_i)-\mu_{k,r}^{Green}|,|F_{lr}^{Blue}(x_i,y_i)-\mu_{k,r}^{Blue}|\right]<\lambda_1\sigma_{k,r} \wedge F_{lr}^{Red,Green,Blue}(x_i,y_i)\neq 0, \quad (3.13)$$

where $F_{lr}^{Red}(x_i,y_i)$, $F_{lr}^{Green}(x_i,y_i)$, $F_{lr}^{Blue}(x_i,y_i)$ are the red, green, and blue components of the pixel, respectively, $\mu_{k,r}^{Red}$, $\mu_{k,r}^{Green}$, $\mu_{k,r}^{Blue}$ are the means of the k^{th} Gaussian model of the pixel color components, $\lambda_1 = 2$ is a hysteresis based variance threshold, and $\sigma_{k,r}$ is the standard deviation of the k^{th} model. We stop matching when the condition is met and update the components using

$$\begin{aligned}
\mu_{k,r}^{Red} &= \mu_{k,r-1}^{Red}+\alpha[F_{lr}^{Red}(x_i,y_i)-\mu_{k,r-1}^{Red}]; && (3.14)\\
\mu_{k,r}^{Green} &= \mu_{k,r-1}^{Green}+\alpha[F_{lr}^{Green}(x_i,y_i)-\mu_{k,r-1}^{Green}]; && (3.15)\\
\mu_{k,r}^{Blue} &= \mu_{k,r-1}^{Blue}+\alpha[F_{lr}^{Blue}(x_i,y_i)-\mu_{k,r-1}^{Blue}]; && (3.16)\\
\sigma_{k,r} &= \sigma_{k,r-1}+\alpha\left[max|F_{lr}^{Red,Green,Blue}(x_i,y_i)-\mu_{k,r}^{Red,Green,Blue}|-\sigma_{k,r-1}\right] && (3.17)\\
\omega_{k,r} &= \omega_{k,r-1}+1. && (3.18)
\end{aligned}$$

where $\alpha = 0.005$ is the update factor. If the pixel does not match with any of the already formed classes, it is used as the mean for a new class that replaces the least probable class. To reduce computations, hysteresis use is limited to the situation when a match with the first background component is found. Meaning, we update the parameters of the background or first component subject to the following condition

$$max\left[|F_{lr}^{Red}(x_i,y_i)-\mu_{k,r}^{Red}|,|F_{lr}^{Green}(x_i,y_i)-\mu_{k,r}^{Green}|,|F_{lr}^{Blue}(x_i,y_i)-\mu_{k,r}^{Blue}|\right]<\lambda_2\sigma_{1,r} \\ \wedge\ F_{lr}^{Red,Green,Blue}(x_i,y_i)\neq 0 \quad (3.19)$$

where $\lambda_2 = 1$ is the second hysteresis based variance threshold. We initialize the means of the first model to the observation F_l and impose the condition $F_{lr}^{Red,Green,Blue}(x_i,y_i) \neq 0$ to avoid the weights of black pixels around the border of the compensated frames from exceeding those of actual pixels and thus appearing in the background model. This is an advantage over using the median as black pixels around the border move from being outliers rejected by the median to being part of the background model. This situation happens for large neighborhood sizes, which is needed for slow moving objects. Another advantage is related to the progressive addition of compensated frames. If we judge the quality of the produced background model to be insufficient and require adding more compensated frames, we use the same already developed mixture of Gaussians model and update it with the new observations.

After the background model G_l is generated, we use it with the current frame F_l for color based segmentation as shown in Fig. 3.9. First, we produce the difference frames D_l^Y, D_l^U, and D_l^V in each of the three color channels Y, U and V. Next, we use color information in D_l^U and D_l^V to compensate the gray-level differences in D_l^Y for cases when the luminance value of the object surface is close to that of the background. Finally, we threshold the compensated D_l^Y. For thresholding, we begin by applying an empty frame detection method. If the frame is empty, we use significance thresholding to calculate an empty-frame threshold, denoted t_l^{Φ}, to binarize the frame. If the frame contains regions of change (ROC), we calculate separate thresholds for ROC blocks, denoted t_l^r, and non-ROC blocks, denoted t_l^b. Local block-based t_l^b and t_l^r thresholds are then averaged to produce the global threshold, denoted t. Because of propagated motion estimation errors, D_l^Y contains more errors than is normally found in background subtraction with fixed cameras. This tends to raise t and causes erroneous white pixels in B_l. We compensate for this increase in t by multiplying it with an experimental gain $\alpha = 0.7$ and use the adjusted value to binarize the color-compensated difference frame D_l^Y and produce the binary frame B_l. More details on the color based background subtraction and the adaptive thresholding techniques are found in [42, 49], respectively.

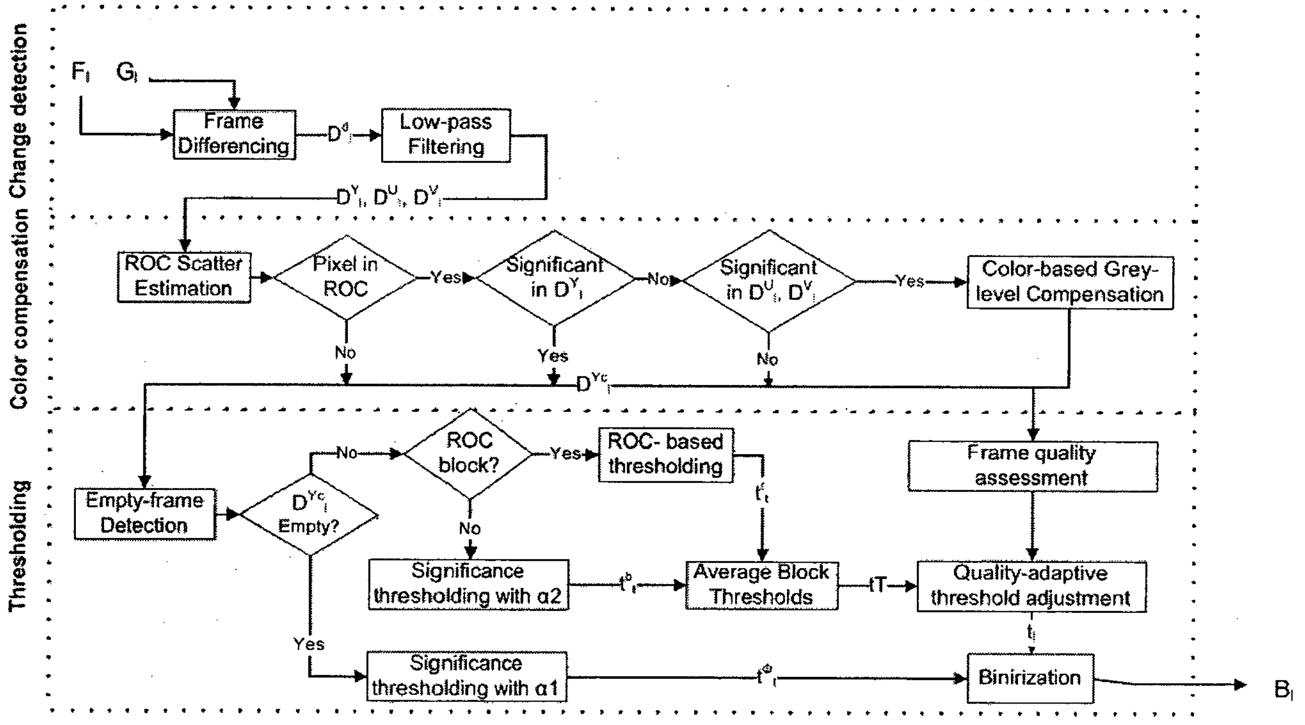


Figure 3.9: Proposed segmentation by integrating color change detection followed by global thresholding with MoG background generation.

Due to the errors from motion estimation, B_l contains many erroneous small blobs. Another motivation for using color background subtraction instead of the gray-level background subtraction used for fast processing is that when using the gray-level background subtraction technique, the maximum filter connects these blobs forming larger errors. These large errors are misclassified as object pixels. The color background subtraction technique does not use a maximum filter, but because of color compensation leads to less holes and gaps inside the objects. We post process B_l using morphological opening, closing, and by holes filling to remove the errors from the final output of color segmentation. Noise can interfere with the global motion estimation process leading to inaccurate global motion parameters. This leads to misaligned compensated frames and to eventually reduces the quality of the background model. While we use a global motion estimation method [63] which is more robust to noise than other GME methods (e.g., [62]), the propagation of motion parameters from short term to long term parameters aggravates the errors.

3.5 Results

3.5.1 Parameters and Limitations

For the fast segmentation method, we use $g_{lim} = 63$ and quantization levels 80, 120, and 200 for thresholding [68], which produce temporally stable objects for tracking in normal illumination and contrast conditions.

For the segmentation with global motion, we use a maximum neighborhood size of $N_r = 10$. Increasing N_r reduces the quality of the generated background models due to accumulated estimation and compensation errors. Lowering it is not enough to completely remove objects from the background model. Both the proposed and reference approaches rely on detecting the background behind the objects as they move. When these objects move very slowly or remain static, they essentially become part of the background and both algorithms cannot detect the details behind them. Relying on spatial techniques like inpainting can help solve this problem.

3.5.2 Measuring Accuracy

We measure accuracy using the statistical F-measure. It incorporates both precision and recall. Precision is defined as the number of relevant results retrieved by a search divided by the total number of results retrieved by that search. Recall is defined as the number of relevant results retrieved by a search divided by the total number of existing relevant results. Precision and recall are calculated using

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \tag{3.20}$$

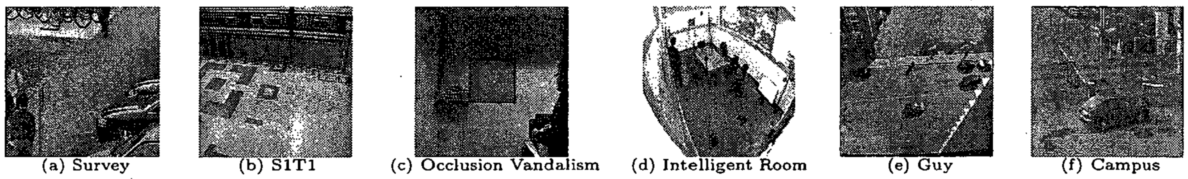
where TP is true positive, and FP is false positive, and FN is false negative. F-measure is calculated from precision and recall using

$$\text{F-measure} = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}. \tag{3.21}$$

It is used to compare the segmentation output of a method to that of another method or to the ground truth segmentation to evaluate its accuracy [6].

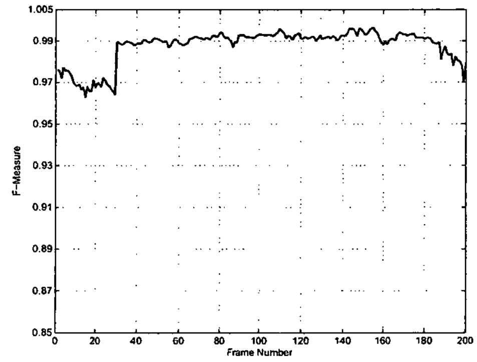
3.5.3 Segmentation with fixed camera

The performance of the fast segmentation method with a fixed background is evaluated in [68] and the performance of the background update technique in [5]. Integrating both methods removes the restrictions of a fixed background from [68] by updating the background. We verify that the performance of [68] is retained after integrating the background update technique using the statistical F-measure. We use the F-measure to compare the output of the fast segmentation method with the ground truth background to that with the automatically generated background model. We use the sequences *Survey*, *S1T1*, *Occlusion Vandalism*, *Intelligent Room*, *Guy*, and *Campus* in Fig. 3.10 for testing segmentation with fixed camera. The F-measure comparison results are shown in Fig. 3.11. We also compare the segmentation outputs visually in Figs. 3.12-A.4. The accuracy (F-measure) is on average 96% indicating that the performance of the method in [68] was retained after the background update technique is integrated. This can also be confirmed visually in Figs. 3.12-A.4. In Fig. 3.12, because of temporal averaging in background generation, less errors are produced in the segmentation output for the automatic background case for the *Survey* sequence. In Fig. 3.13, the automatically generated background in the last frame produce less false negatives for the far object for the *S1T1* sequence. It also produce less false positives for the *Vandalism Occlusion* sequence in Fig. A.1 in the appendix. The results for the sequences *Intelligent Room*, *Guy*, and *Campus* in Figs. A.2, A.3, and A.4 in the appendix are visually similar. In Chapter 5, we demonstrate the benefits of integrating the background update technique from the point of view of tracking and event detection.

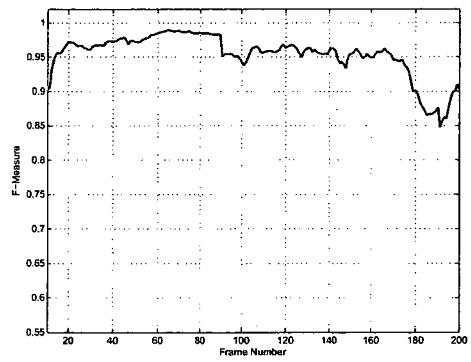


(a) Survey (b) S1T1 (c) Occlusion Vandalism (d) Intelligent Room (e) Guy (f) Campus

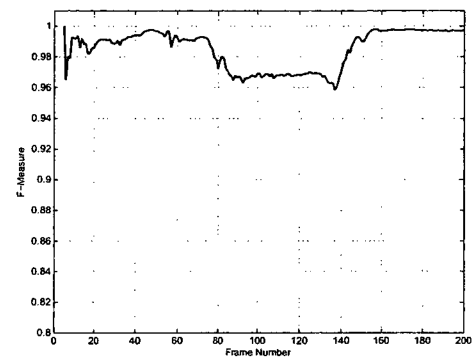
Figure 3.10: Video sequences used for testing segmentation with fixed camera.



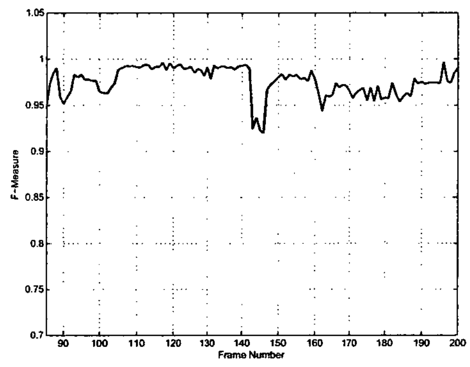
Survey



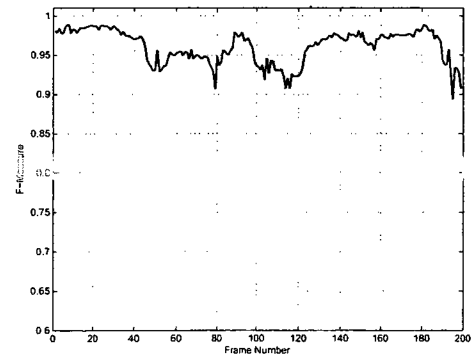
S1T1



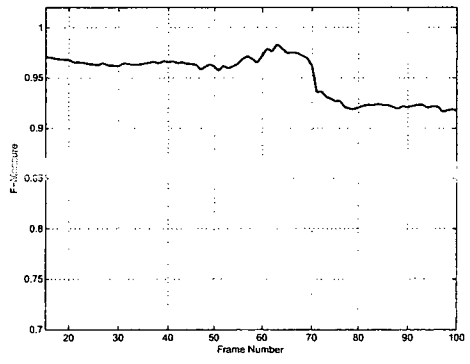
Occlusion Vandalism



Intelligent Room



Guy



Campus

Figure 3.11: Objective comparison of the fast segmentation method with the ground truth background compared to the segmentation with the automatically generated background model using [5]. The accuracy is on average 96% indicating the performance of the fast segmentation method is retained after its integration with the background update method.

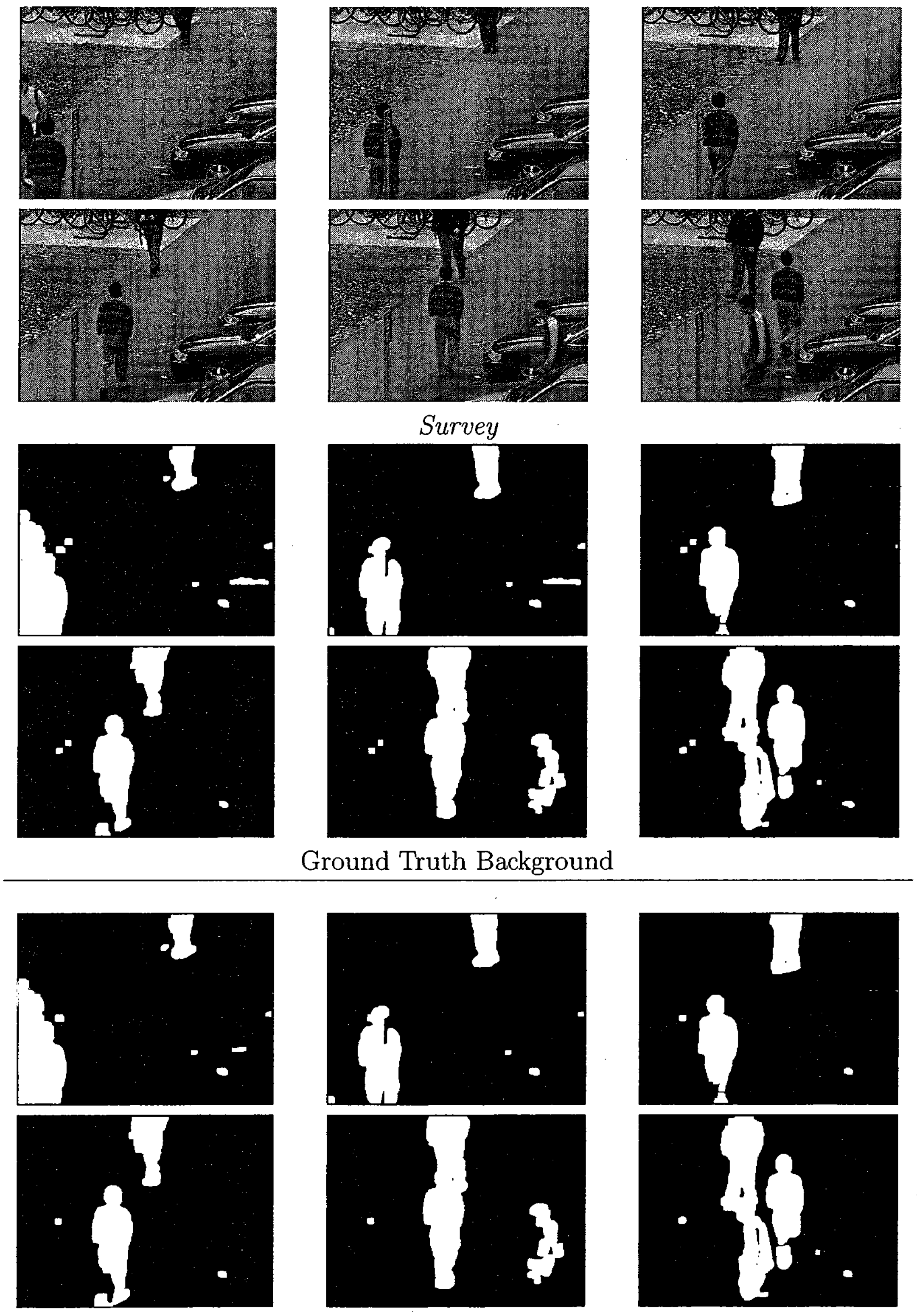


Figure 3.12: Subjective comparison of the fast segmentation method with ground truth and automatically (using [5]) generated background models for frames $l = 0$ to 100 of the *Survey* sequence.

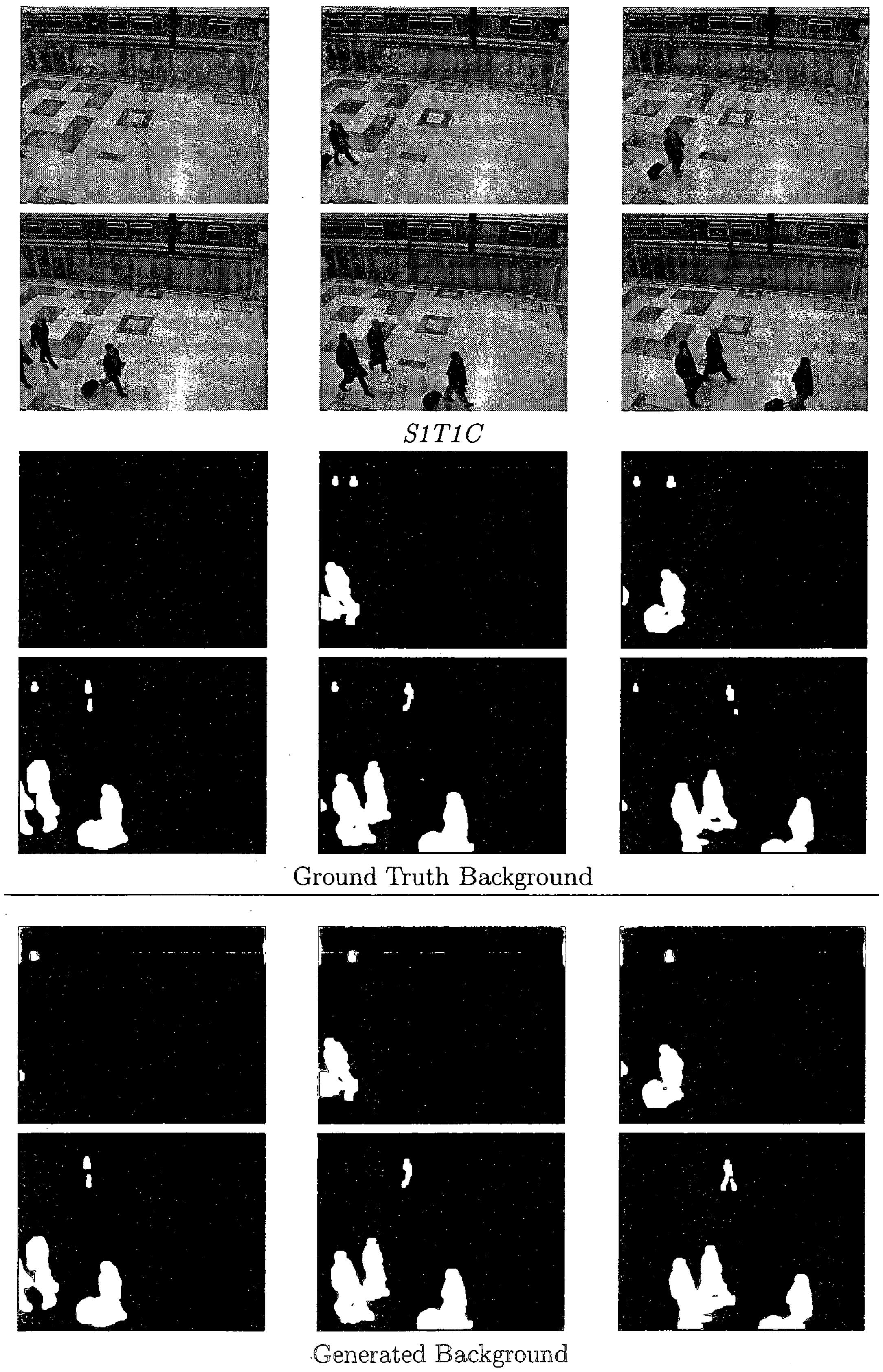


Figure 3.13: Subjective comparison of the fast segmentation method with ground truth and automatically (using [5]) generated background models for frames $l = 0$ to 100 of the *S1T1* sequence.

3.5.4 Segmentation with global motion

We use the eleven video sequences in Fig. 3.14 to evaluate the performance of the proposed segmentation method with global motion. The *Stefan* sequence shows a tennis player. We show the results for the part of the video sequence in which the object (tennis player) moves very fast to hit the ball. The sequences *Lab* and *Guy Zoom* show pure translation and pure zoom motions, respectively. The sequence *Mountain* is part of a BBC documentary and shows a leopard in fast irregular camera motion. The sequence *Race* shows kart racing with the objects moving towards the camera. The sequence *Football* shows a game of American football with fast tracking motion. The sequence *Biathalon* shows an object in a high texture environment. The sequences *Figure Skating* and *Soccer* show objects in homogeneous environments. The sequences *Formula 1* and *Hockey* show fast pan camera motion. Note that sequences like *Mountain*, *Football*, *Figure Skating*, *Soccer*, *Biathalon* are extracted from compressed video sequences.

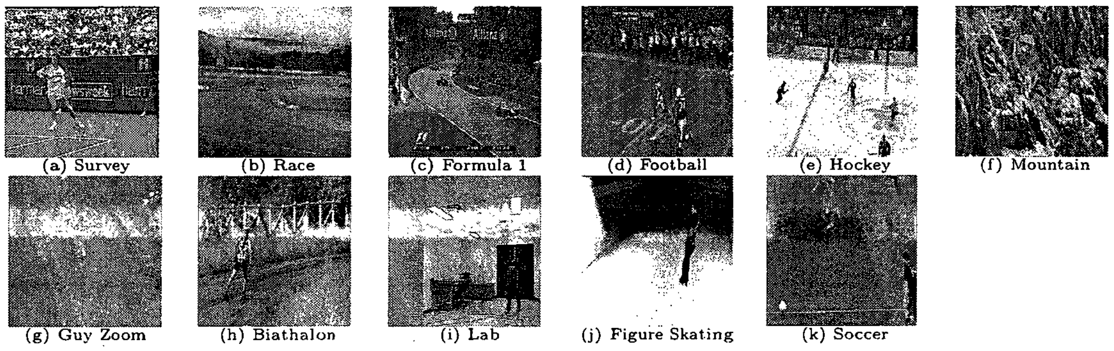


Figure 3.14: Video sequences used for testing segmentation with moving camera.

We can separate the process of segmentation with global motion to two phases. The first is background generation followed by the second which is segmentation. We start by fixing the segmentation method to that of [6] and compare the segmentation outputs obtained using the backgrounds generated by proposed and reference [6] methods. The results in Fig. 3.15 show that the proposed background generation method improves the segmentation accuracy.

We now fix the background generation method to the proposed one and vary the segmentation technique. The results are shown in Fig. 3.16. The proposed segmentation method retains or exceeds the accuracy of the reference segmentation method for all sequences except the *Guy Zoom* sequence due to the similarity between the street and cars colors and low contrast, while being less computationally complex. We finally compare the segmentation output of using the proposed method and the method in [6] (with [62] for GME) as a whole in Fig. 3.17. The proposed method as a whole retains or exceeds the accuracy of the reference method especially during fast motion (e.g., Stefan), while reducing its complexity in both background generation and segmentation.

We visually compare the backgrounds generated by the proposed and reference methods in Figs. 3.18-3.20. In Fig. 3.18, both the proposed and reference [6] methods produce visually accurate background models. The proposed method handles the frame corners better than the reference method in Fig. 3.19 and causes less blurring as seen in the *Football* sequence. In Fig. 3.20, the proposed method handles the fast motion in the *Formula 1* sequence better than the reference method. It also generates a cleaner background model for the *Figure Skating* sequence.

We visually compare the segmentation outputs in Figs. 3.21-A.7. The segmentation output for the *Stefan* video sequence in Figs. 3.21, 3.22 shows the proposed method produces more connected regions than the reference method. It also produces less false positives. Both the proposed and reference methods produce accurate object masks for the *Race* sequence with slow camera motion in Figs. 3.23, 3.24, however, the proposed method handles fast motion better (generates less false positives) than the method in [6] as seen in Figs. 3.25,3.26 for *Formula 1* sequence, in Figs. 3.27, 3.28 for the *Football* sequence, and in Figs. 3.29, 3.30 for the *Hockey* sequence. Both the proposed and reference methods correctly detect the leopard in the *Mountain* sequence in Figs. 3.31, 3.32, but the proposed method does not produce artifacts around the frame boundaries and corners. Both methods are challenged with the similar colors of the cars and street in the *Guy Zoom* sequence in Figs. 3.33, 3.34, but the proposed method produces less false negatives. In the *Biathalon* sequence in Figs. 3.35, 3.36, the reference method produces less false negatives around the left leg area of the object, but it misses the ski poles which the proposed method detects. In Figs. 3.37, 3.38, the proposed method produces less false negatives than the reference method for the *Lab* sequence. This can also be observed with the *FigureSkating* sequence in Figs. A.5, A.6, and Fig. A.7 with the *Soccer* sequence in the appendix.

A main contribution of the proposed method is reducing the complexity of the reference method in [6], while retaining or exceeding its segmentation accuracy. The complexity is mainly due to repeated background generation with an increasing neighborhood size. Recall that N_r is the maximum neighborhood size and let q be the preliminary neighborhood size. So, we start from q and move our way up to N_r and as we go from one q to the next, the number of frames merged to produce the background model is $2q + 1$ (considering frames before, after and including the current frame). In the worst case scenario, the total number of frames merged in [6] is

$$C = \sum_{q=1}^{N_r} 2q + 1 = 2\sum_{q=1}^{N_r} q + 1 = 2\frac{N_r^2 + N_r}{2} + 1, \tag{3.22}$$

while in the proposed method, we do not need to merge frames from the previous step as they are already included in our model and we just need to supply the background generation step with the additional observations from the current step. Thus, the number of frames merged for the proposed method is $C = 2N_r + 1$. The method in [6] uses the median to produce the background frame which has a complexity of $O(Clog(C))$, while the proposed method uses linear search to find the Gaussian model best matching the new pixel observation which has the linear complexity $O(C)$.

Substituting C and including the frame size N since it is the other factor affecting complexity, the complexity of the technique in [6] is $O(NN_r^2 log(N_r))$ compared to that of the proposed method $O(NN_r)$.

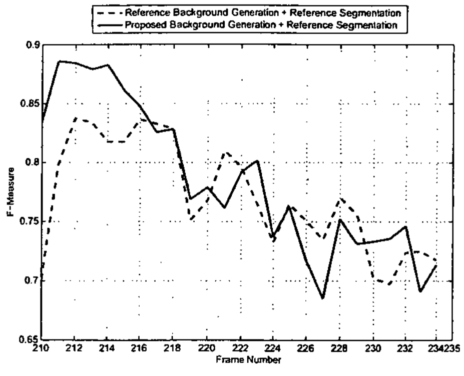
Since we already have the motion parameters describing the motion between F_l and F_{l-1}, and given that consecutive background models are highly similar, another major reduction in complexity comes from alternating between full estimation using neighboring frames for odd l and compensating the motion of B_{l-1} to produce B_l otherwise. This further reduces the complexity. Note that further complexity reduction can be achieved if we use background model compensation more than once in between full estimations, however, this reduces segmentation accuracy on the frame boundaries.

We measured the time needed by the proposed and reference methods to generate the color background model from motion compensated CIF frames (e.g., the *Stefan* in Fig. 3.18) sequence on a Dual-Core AMD Opteron 1000 MHz Processor and Linux. The proposed background generation technique is more than 5 times faster than that of [6] due to eliminating the need for repeated median blending and to alternating between full background model estimation and previous background model compensation. The proposed segmentation is 3.7 times faster than the segmentation of [6].

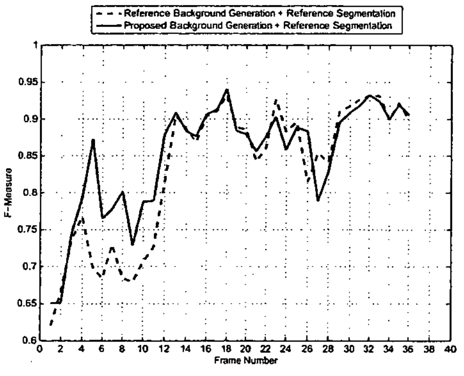
3.6 Summary

This chapter proposed first a fast video object segmentation method with no global motion by integrating a gray-level background subtraction followed by thresholding technique with a mixture of Gaussians background update technique. The proposed method retained on average 96% of the accuracy of the background subtraction technique while eliminating its reliance on a fixed background model.

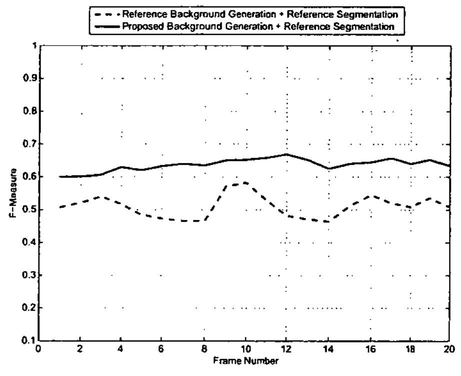
The chapter then proposed a video object segmentation method with global motion. The proposed method starts by using short-term consecutive motion parameters estimated using a segmentation-oriented global motion estimation method to calculate the long-term motion parameters. It then used the long term motion parameters to warp neighboring frames into the coordinate system of the current frame, thus creating multiple observations of the current frame with the objects of interest moving to reveal the background behind them. The proposed method used the observations to estimate the parameters of a mixture of Gaussians model and extract the background model several times faster than the median based approach of a recent method. Further complexity reduction over the recent method was achieved by alternating between full estimation of the background model using multiple observations and motion compensation of the previous frame background model. The generated background models are then used by a color based background subtraction technique followed by adaptive thresholding to produce the binary outputs, which are post-processed using morphological operations to produce the binary outputs. The proposed segmentation with global motion method retained or exceeded the performance of a recent segmentation technique, while being faster in background generation and segmentation.



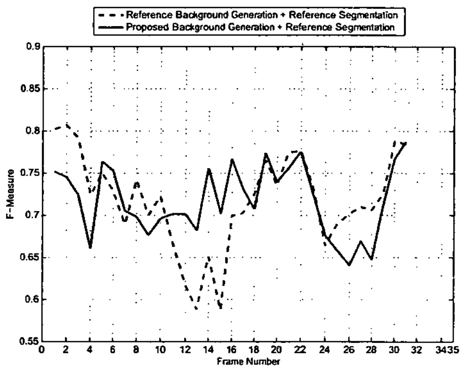
Stefan



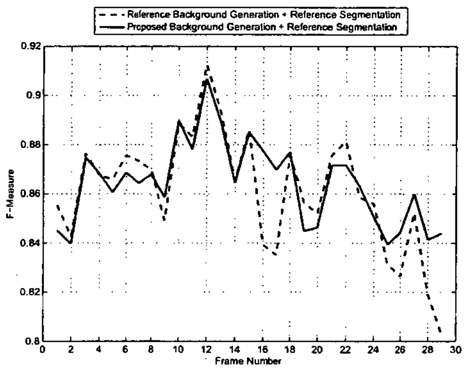
Lab



Guy Zoom



Mountain



Kart Race

Figure 3.15: Objective F-measure comparisons for *Stefan*, *Lab*, *Guy Zoom*, *Mountain*, and *Kart Race* sequences between the proposed and reference background generation techniques using the segmentation of [6].

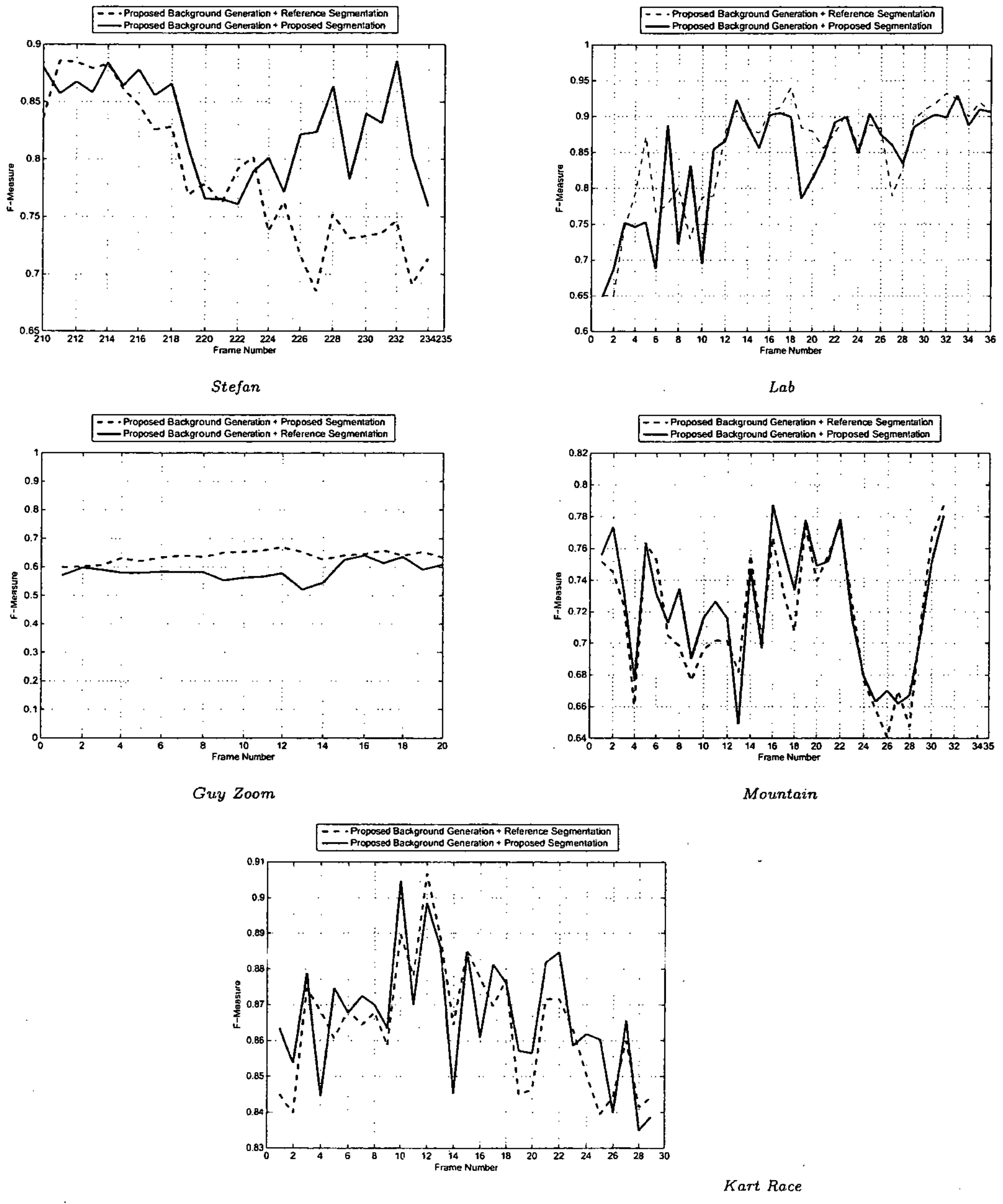


Figure 3.16: Objective F-measure comparisons for *Stefan*, *Lab*, *Guy Zoom*, *Mountain*, and *Kart Race* sequences between the proposed and reference [6] segmentation techniques using the proposed background generation method.

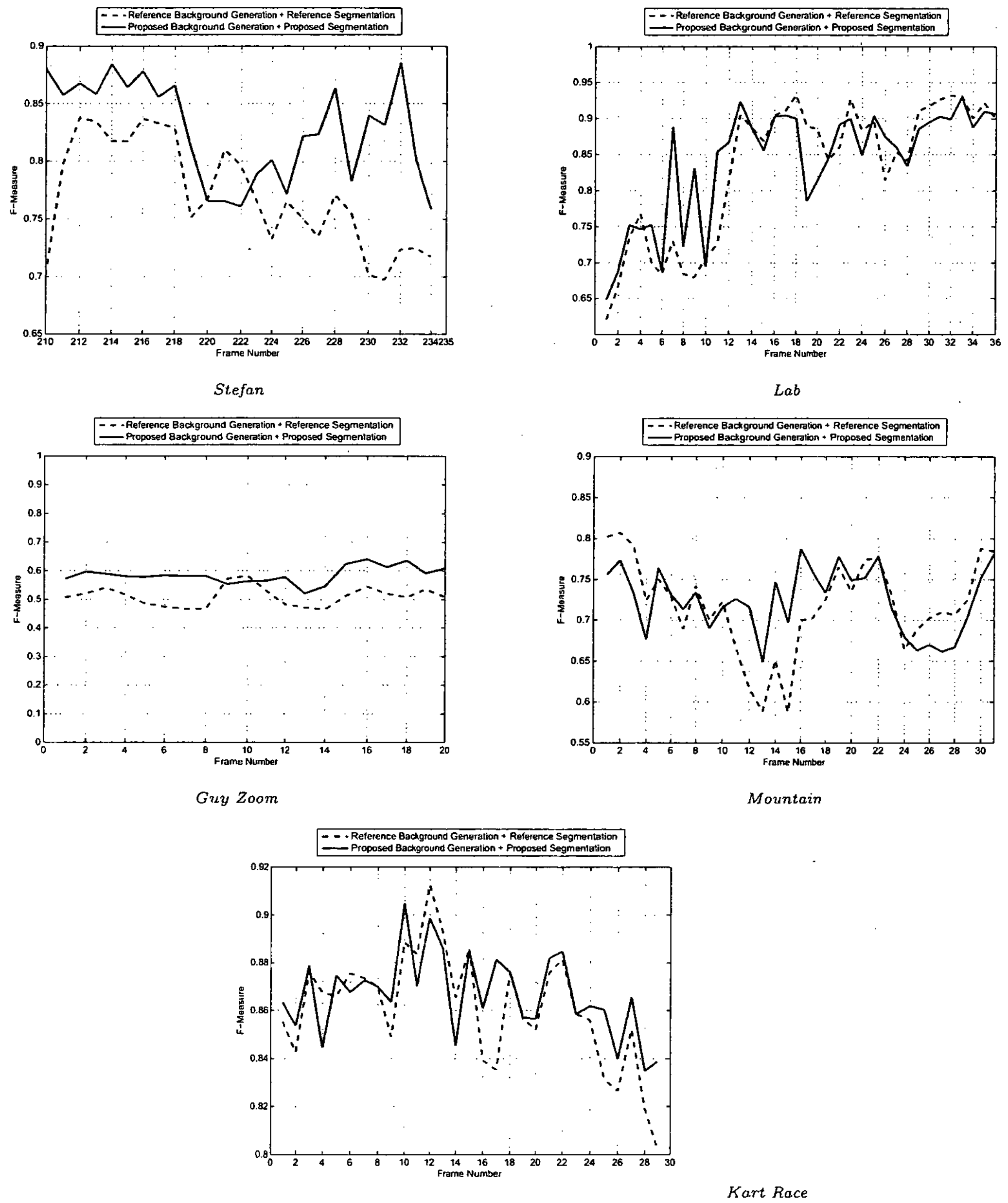


Figure 3.17: Objective F-measure comparisons for *Stefan*, *Lab*, *Guy Zoom*, *Mountain*, and *Kart Race* between the proposed and reference [6] methods.

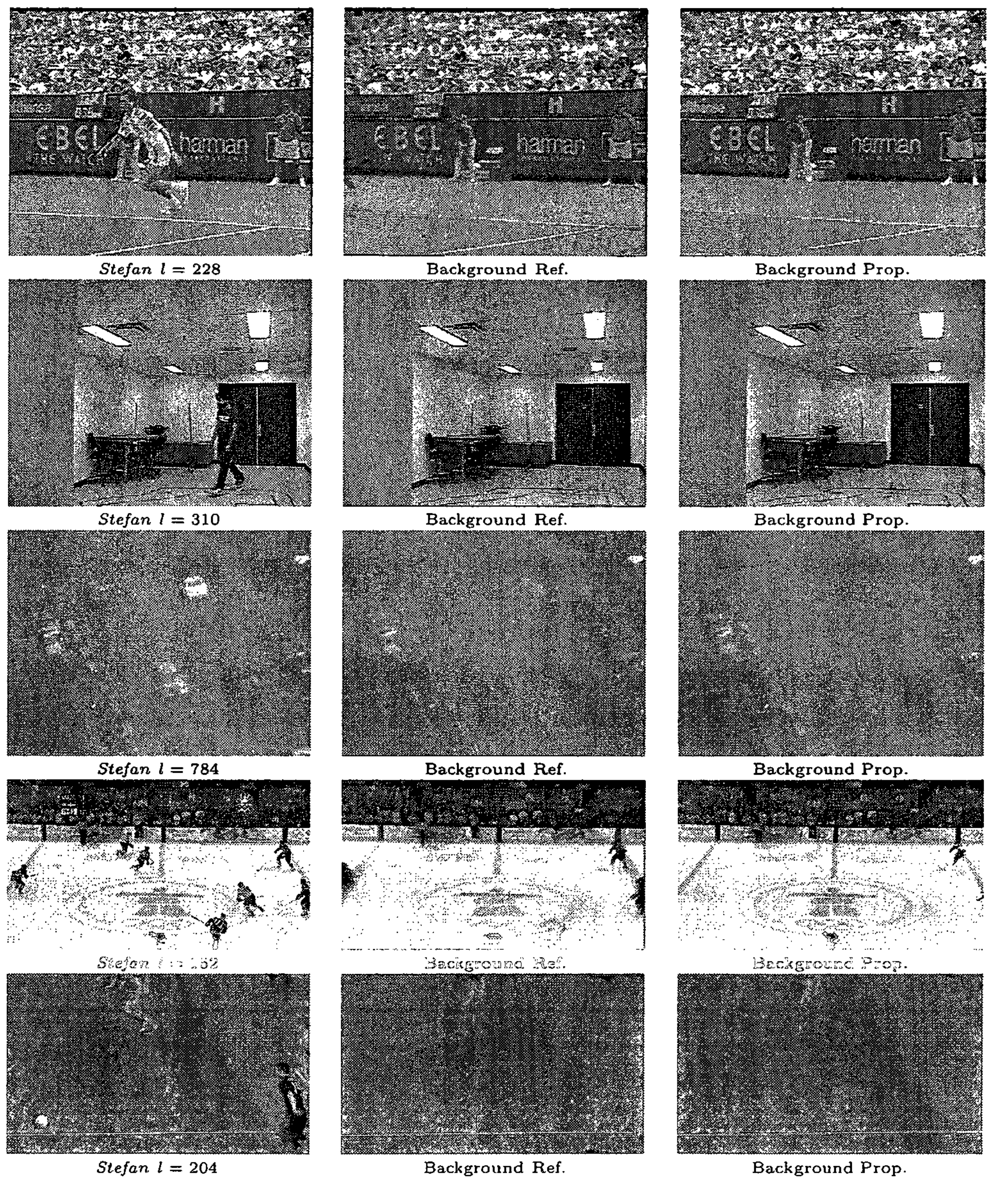


Figure 3.18: Visual comparisons of the generated backgrounds between the proposed and reference methods [6] for the *Stefan*, *Lab*, and *Guy Zoom* sequences

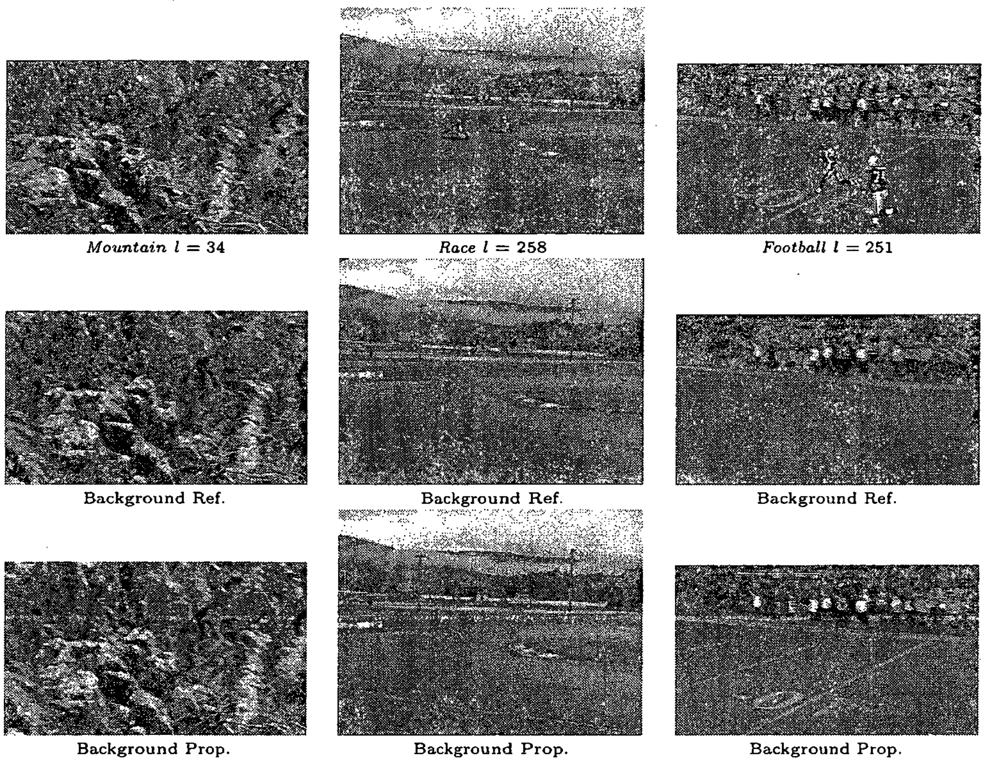


Figure 3.19: Visual comparisons of the generated backgrounds between the proposed and reference methods [6] for the *Mountain*, *Race*, and *Football* sequences

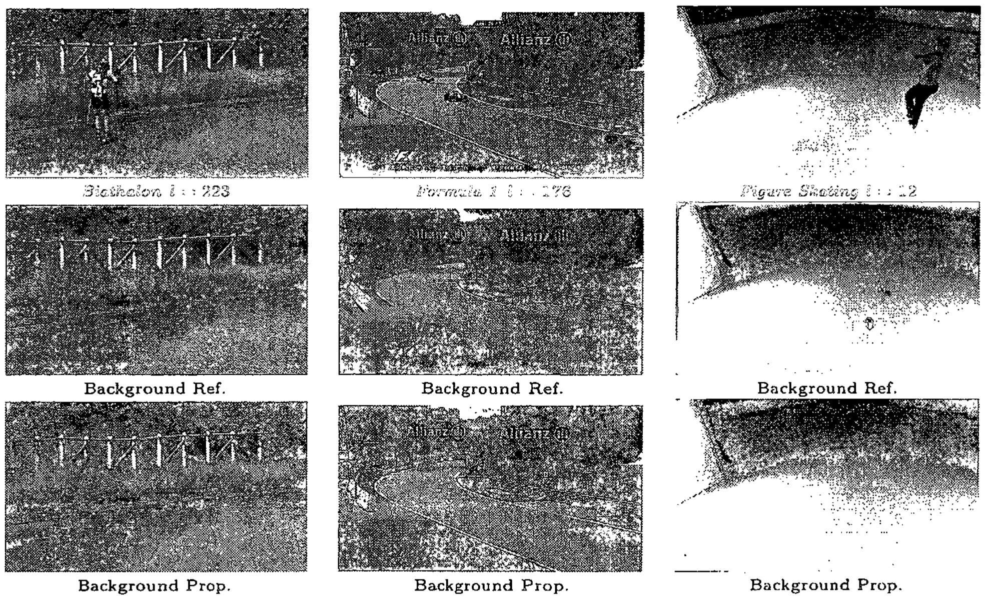


Figure 3.20: Visual comparisons of the generated backgrounds between the proposed and reference methods [6] for the *Biathalon*, *Formula 1*, and *Figure Skating* sequences

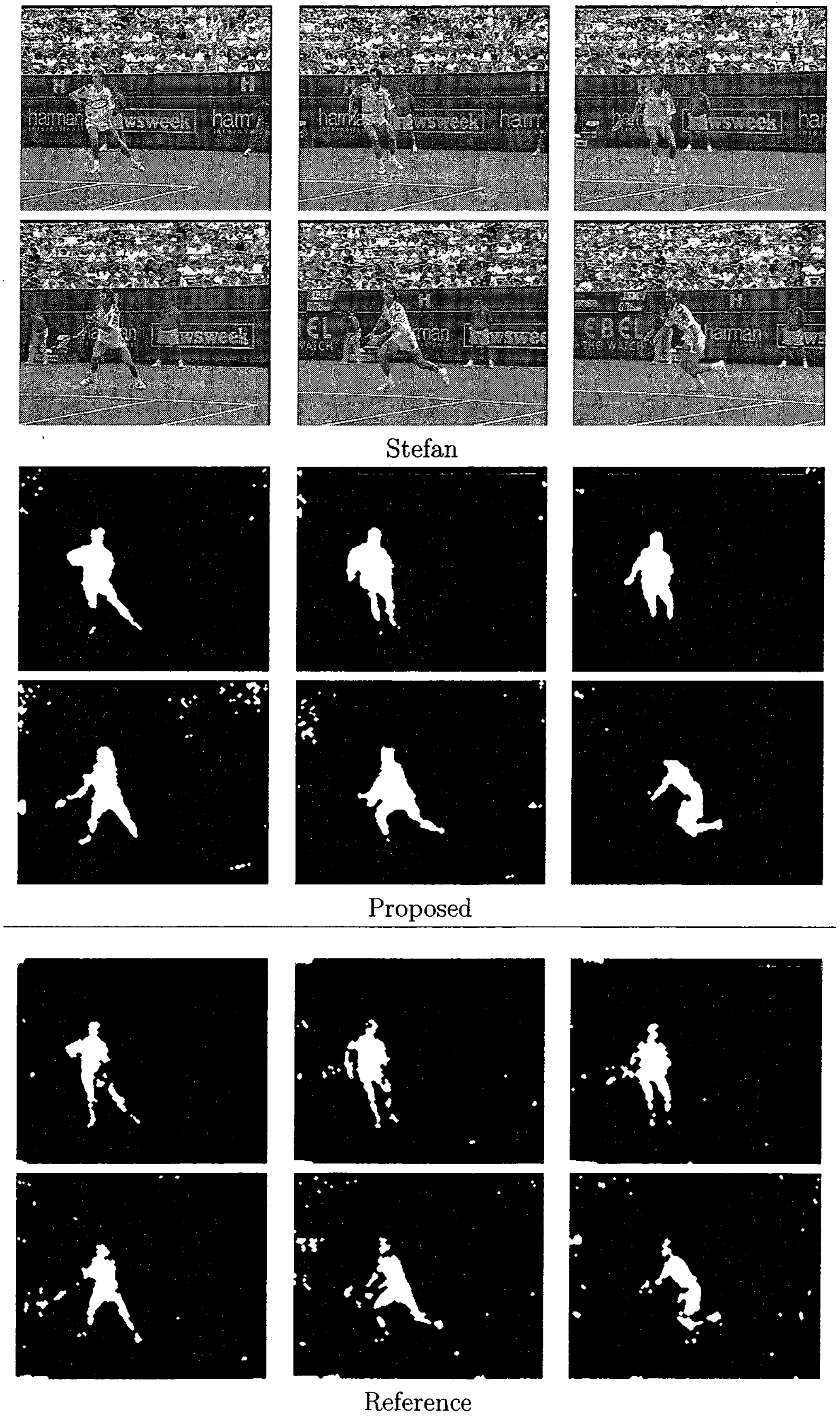


Figure 3.21: Visual comparison of the segmentation output for frames $l = 212$ to 227 of the *Stefan* sequence between the proposed and reference [6] methods.

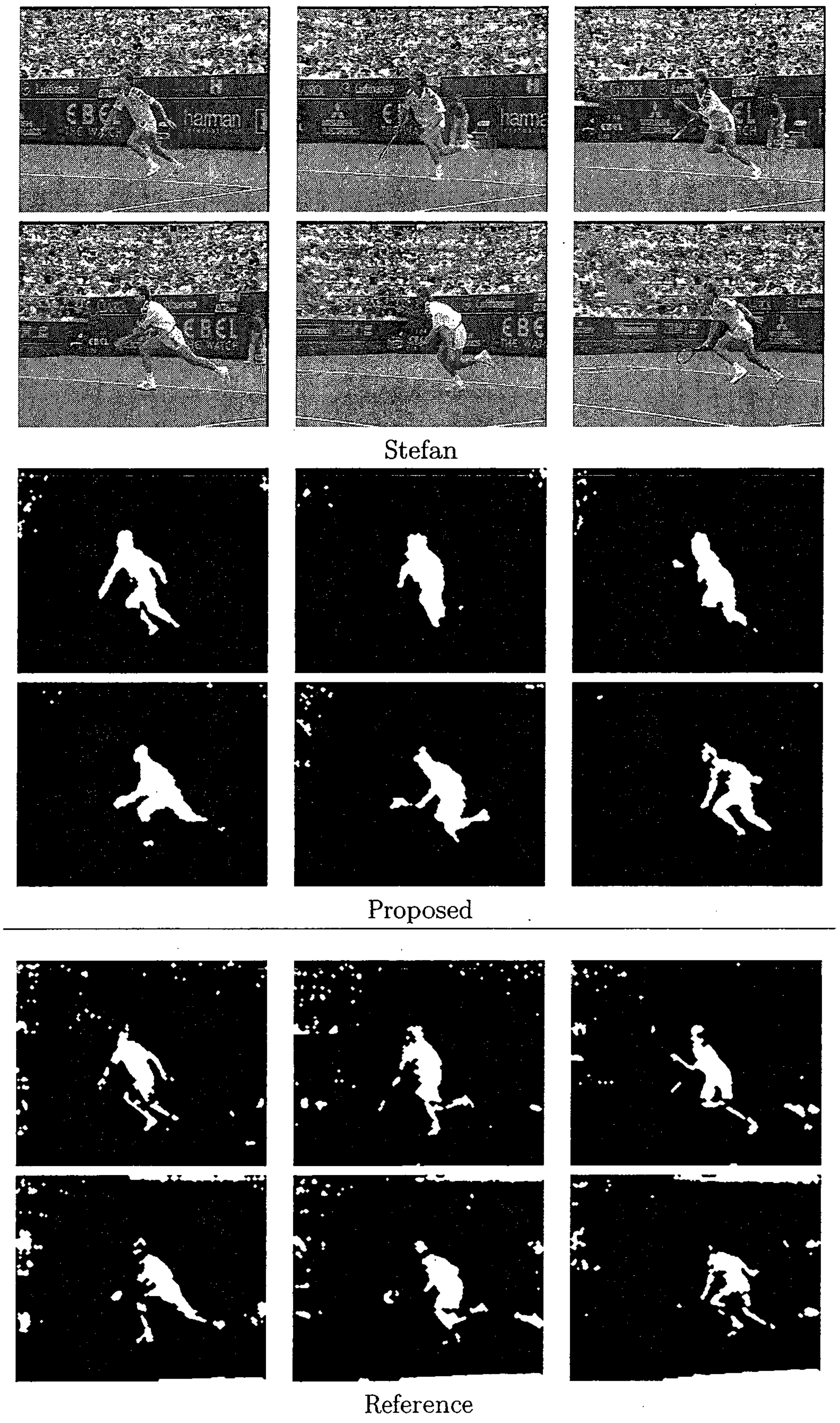


Figure 3.22: Visual comparison of the segmentation output for frames $l = 230$ to 245 of the *Stefan* sequence between the proposed and reference [6] methods.

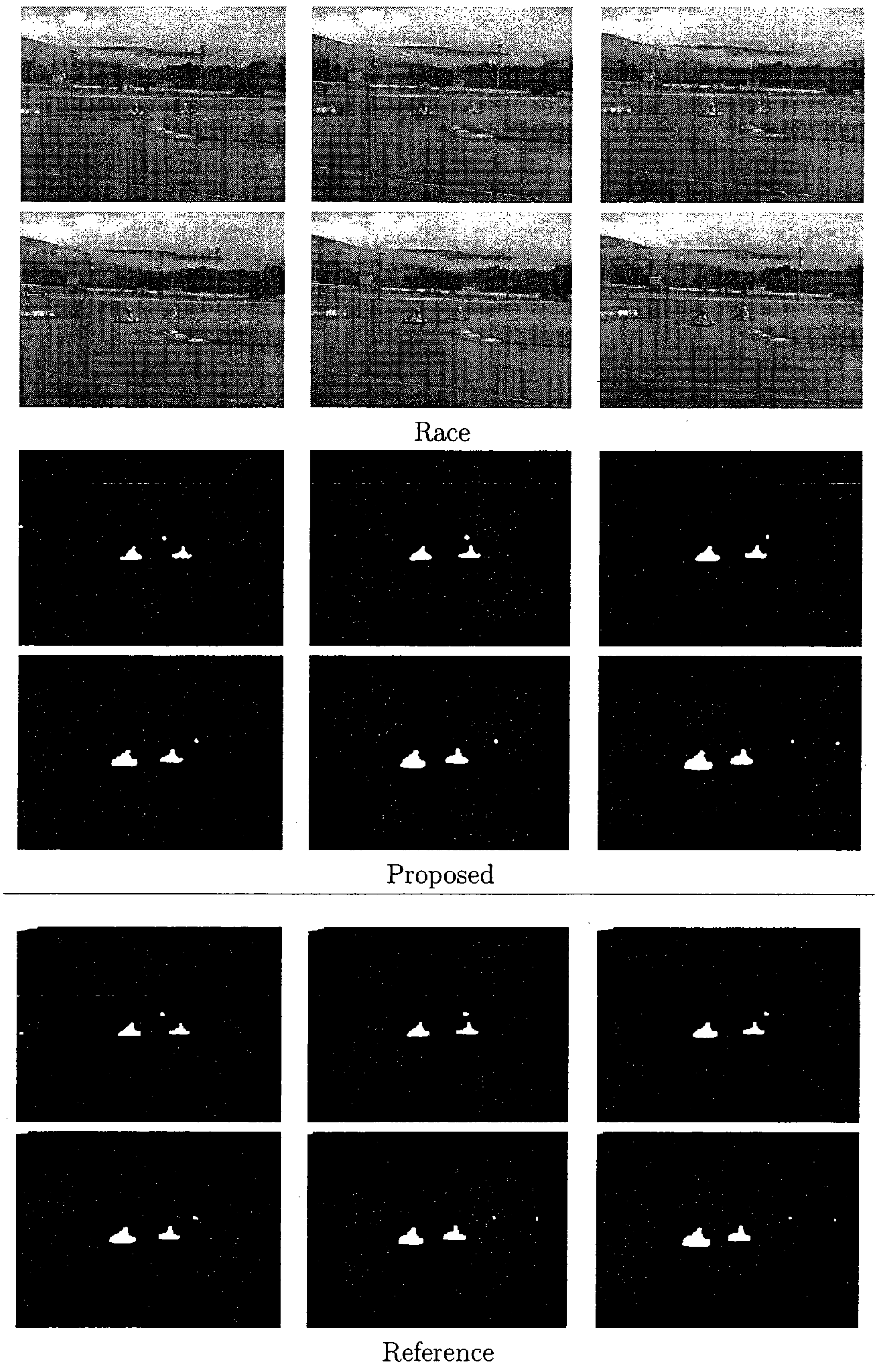


Figure 3.23: Visual comparison of the segmentation output for frames $l = 42$ to 57 of the *Race* sequence between the proposed and reference [6] methods.

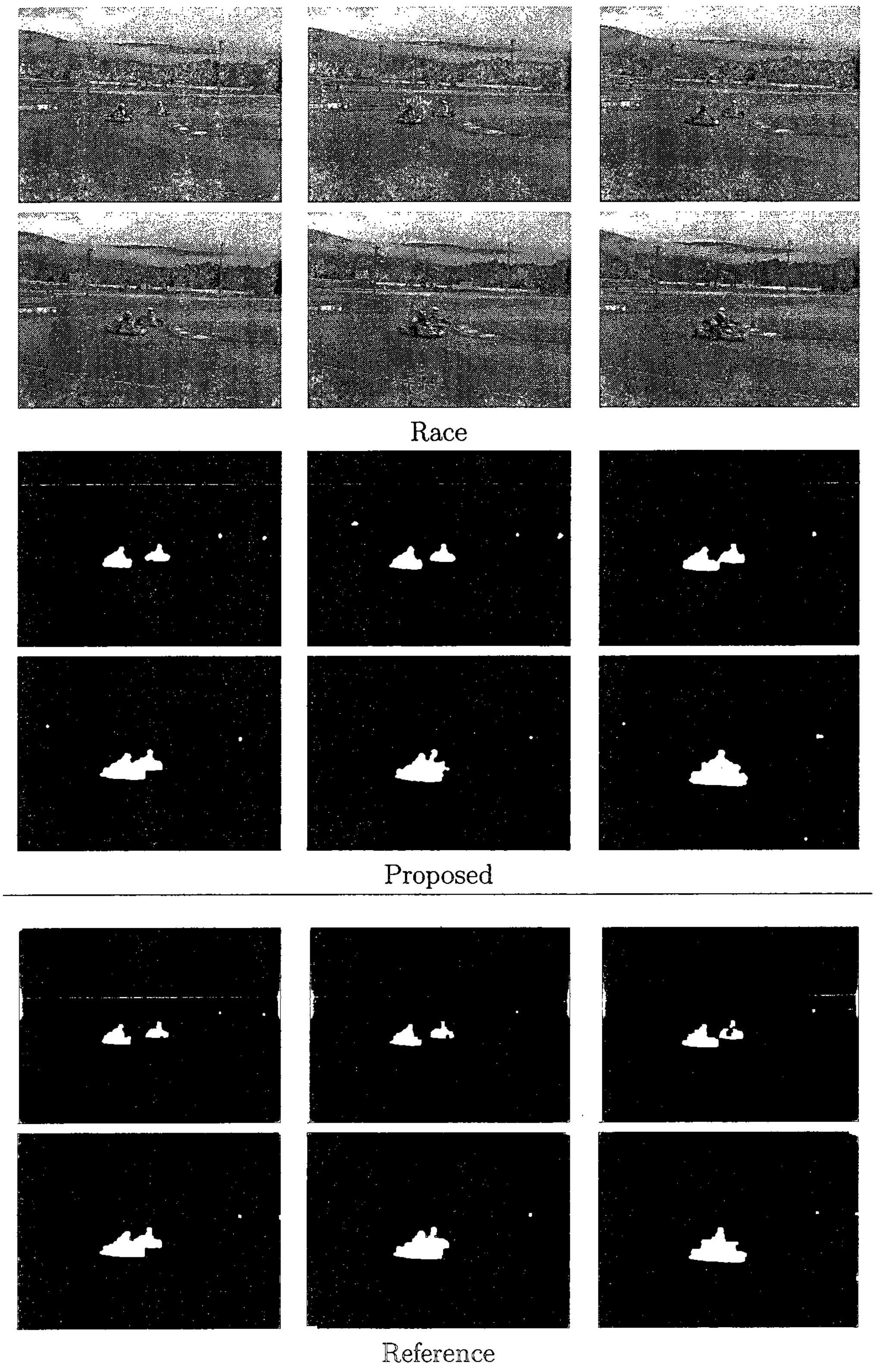


Figure 3.24: Visual comparison of the segmentation output for for frames $l = 60$ to 75 of the *Race* sequence between the proposed and reference [6] methods.

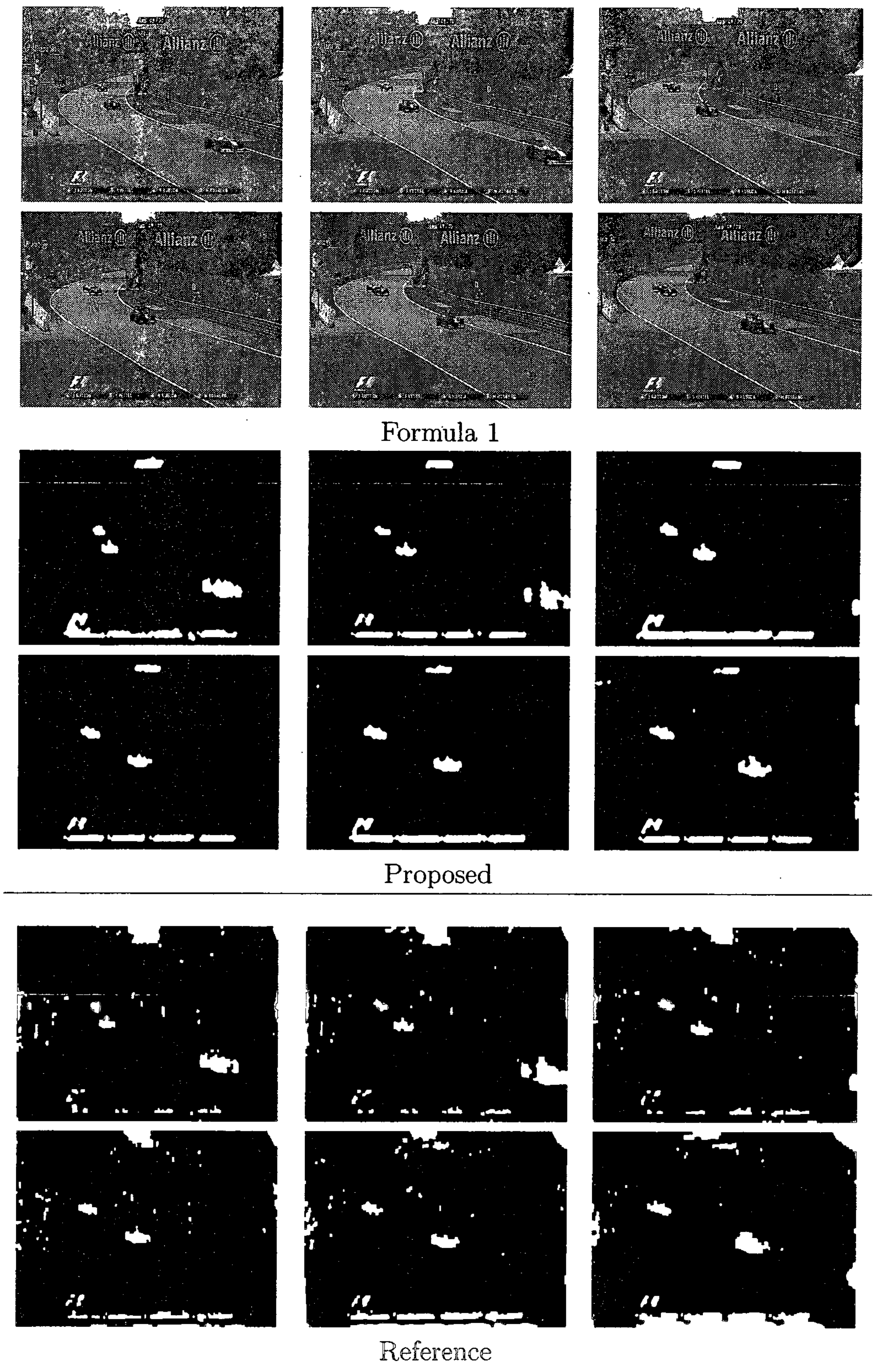


Figure 3.25: Visual comparison of the segmentation output for for frames $l = 12$ to 27 of the *Formula 1* sequence between the proposed and reference [6] methods.

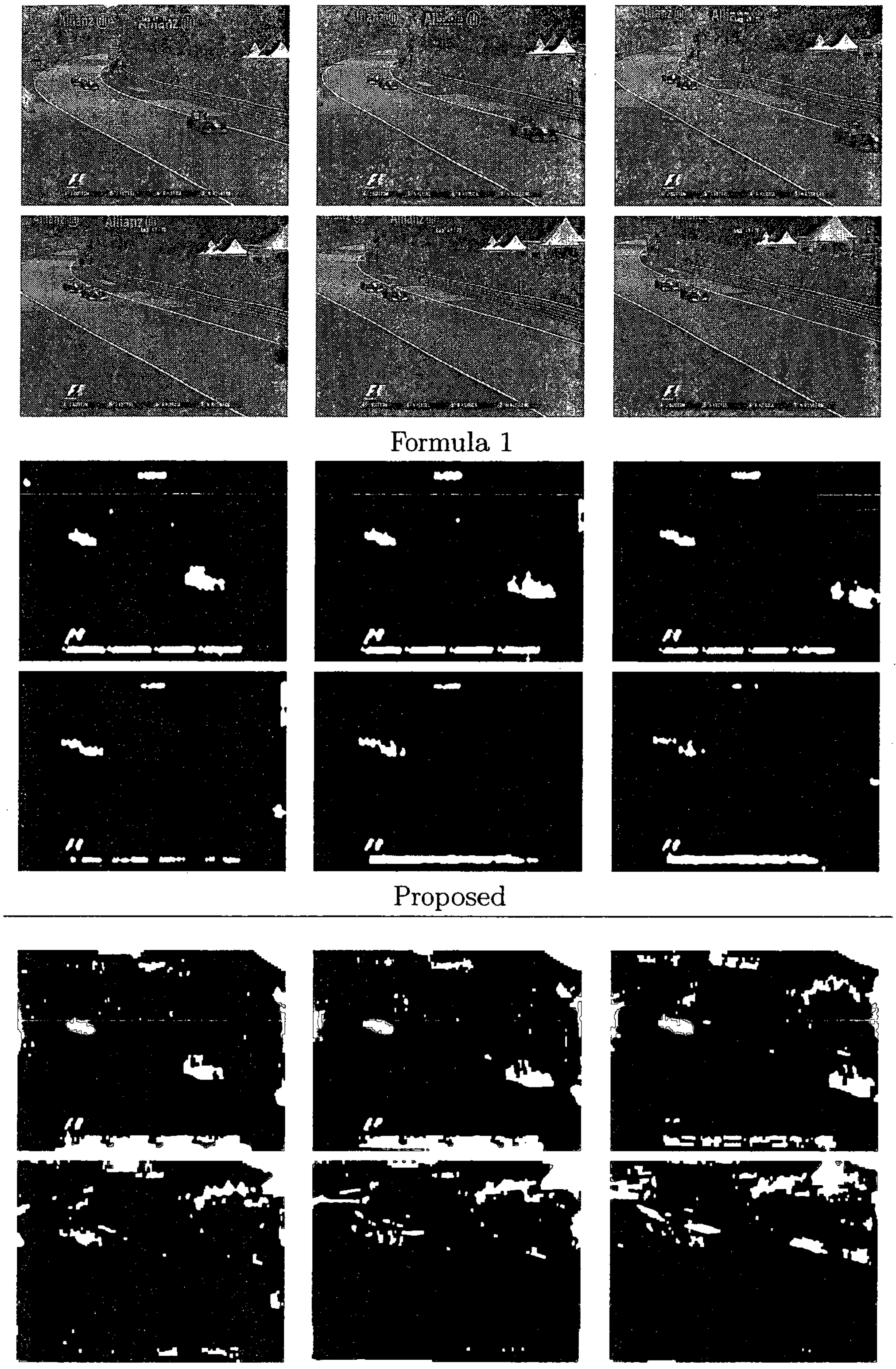


Figure 3.26: Visual comparison of the segmentation output for frames $l = 30$ to 45 of the *Formula 1* sequence between the proposed and reference [6] methods.

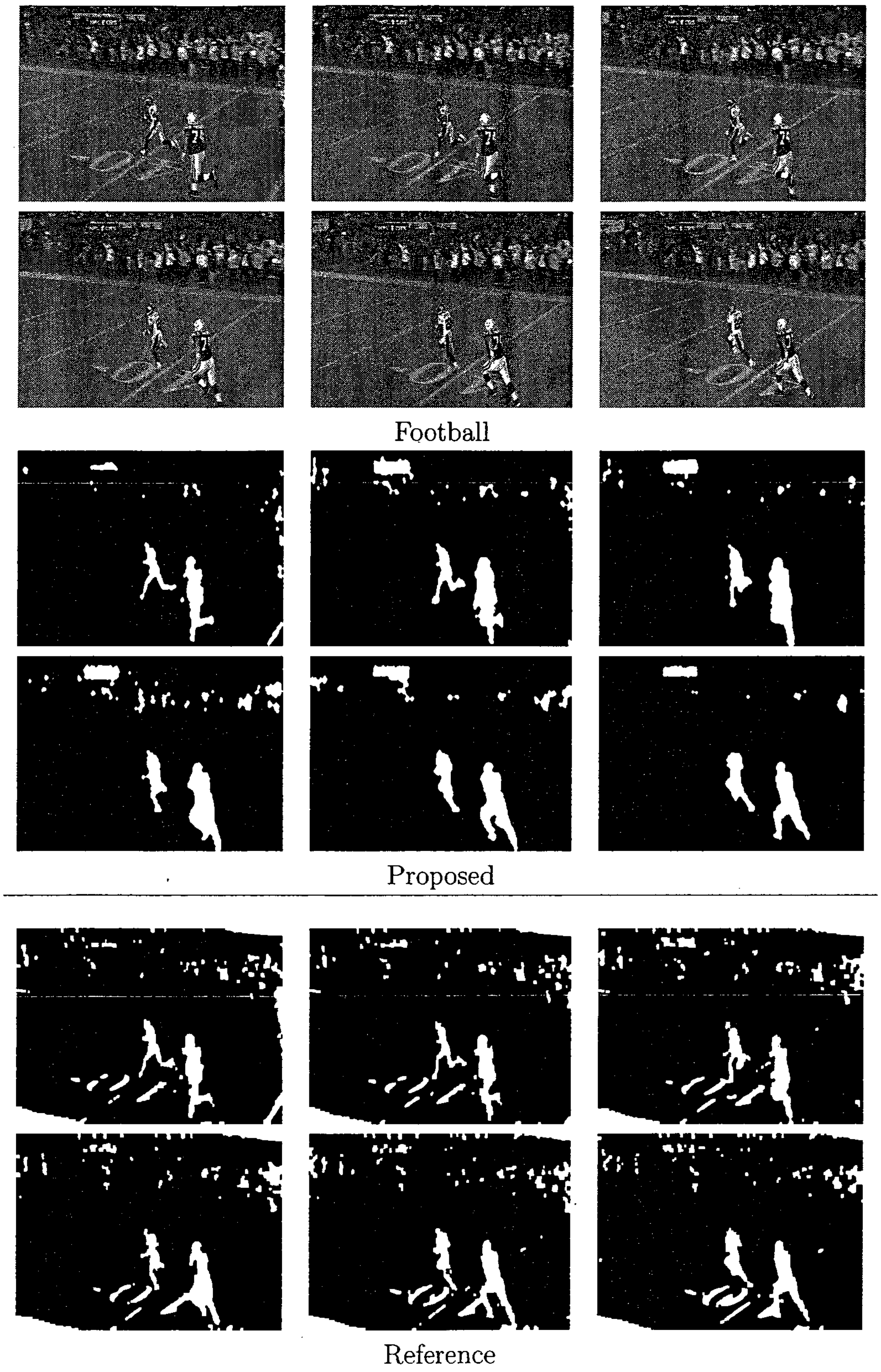


Figure 3.27: Visual comparison of the segmentation output for frames $l = 10$ to 15 of the *Football* sequence between the proposed and reference [6] methods.

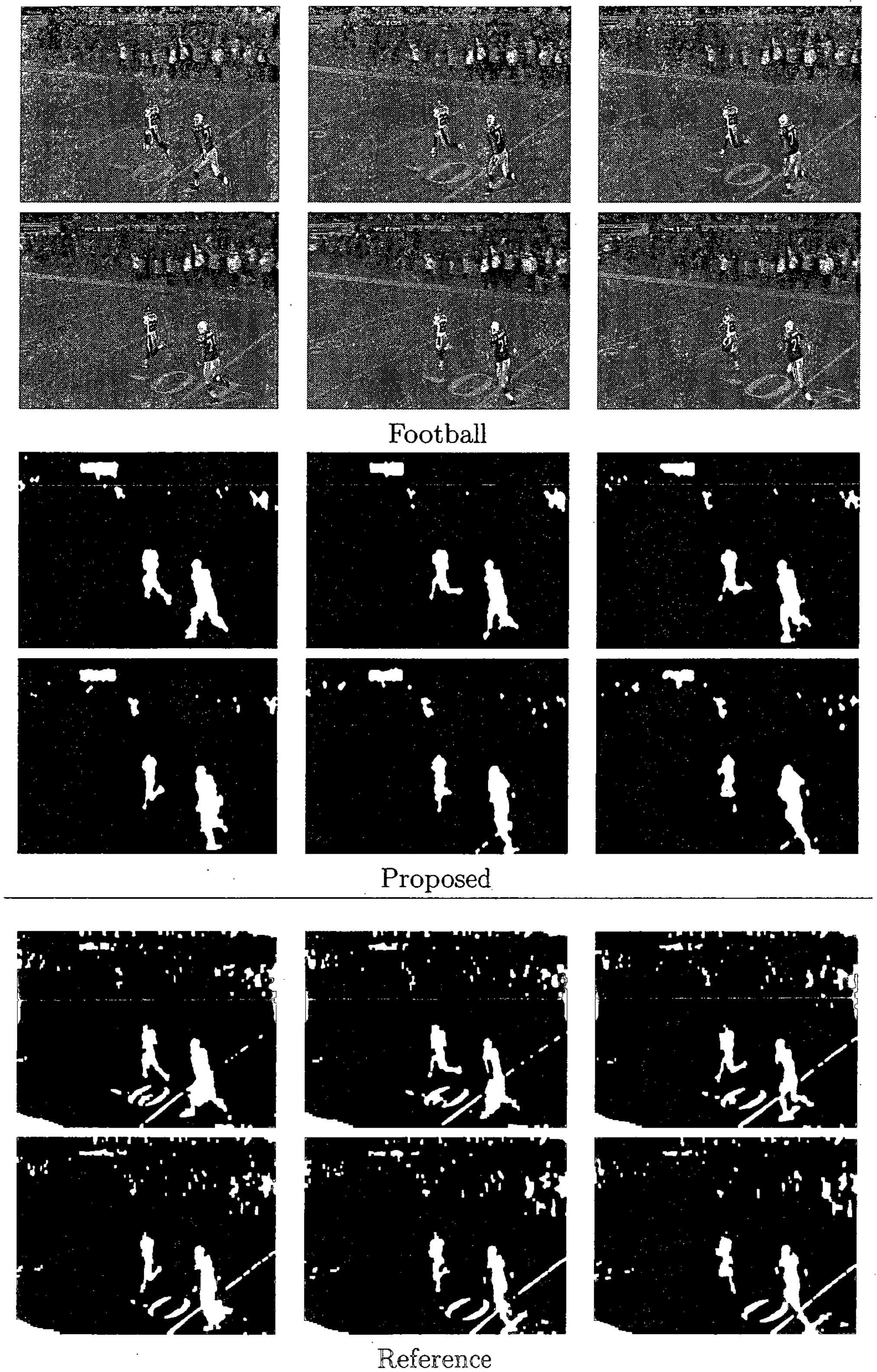


Figure 3.28: Visual comparison of the segmentation output for frames $l = 16$ to 21 of the *Football* sequence between the proposed and reference [6] methods.

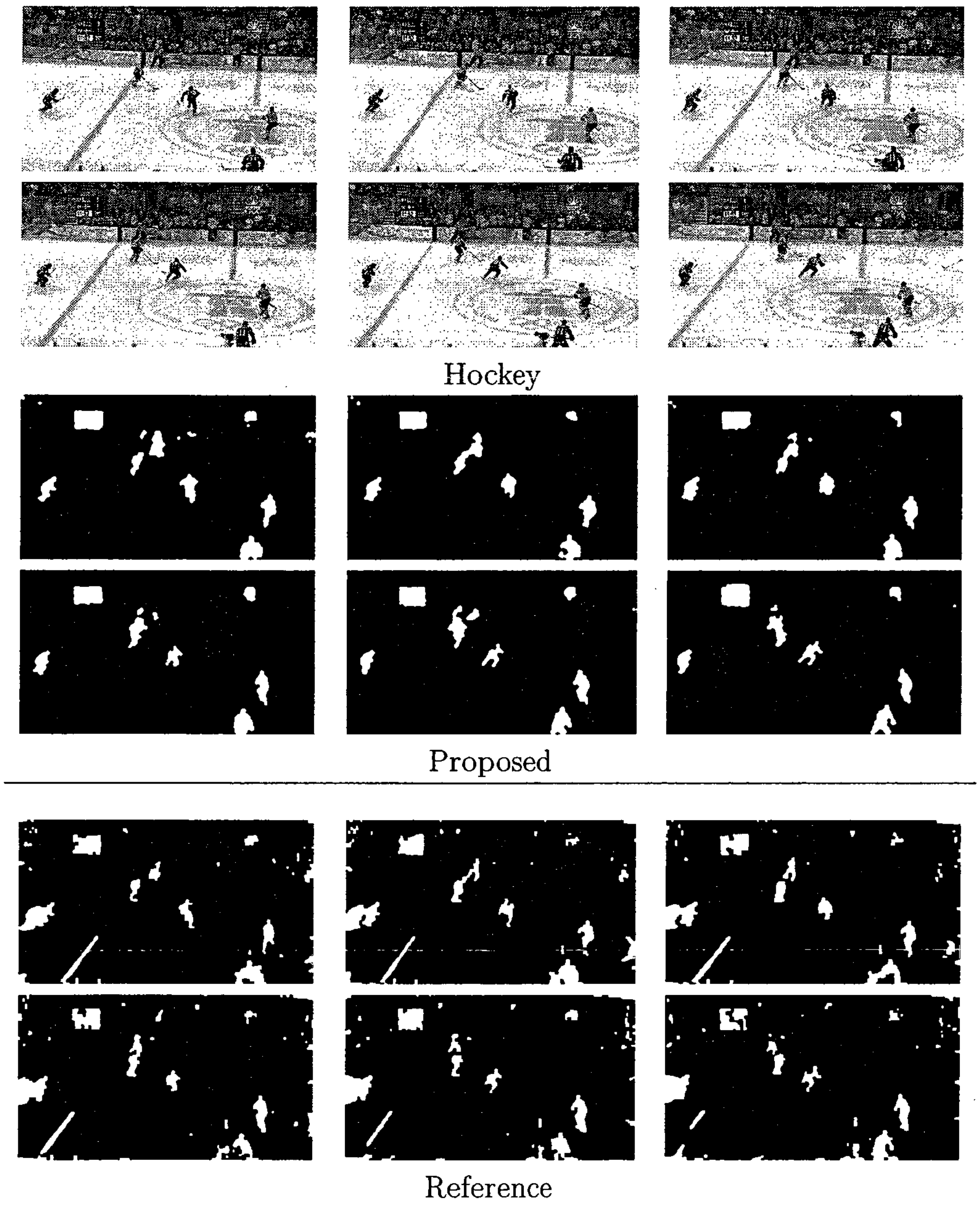


Figure 3.29: Visual comparison of the segmentation output for frames $l = 40$ to 50 of the *Hockey* sequence between the proposed and reference [6] methods.

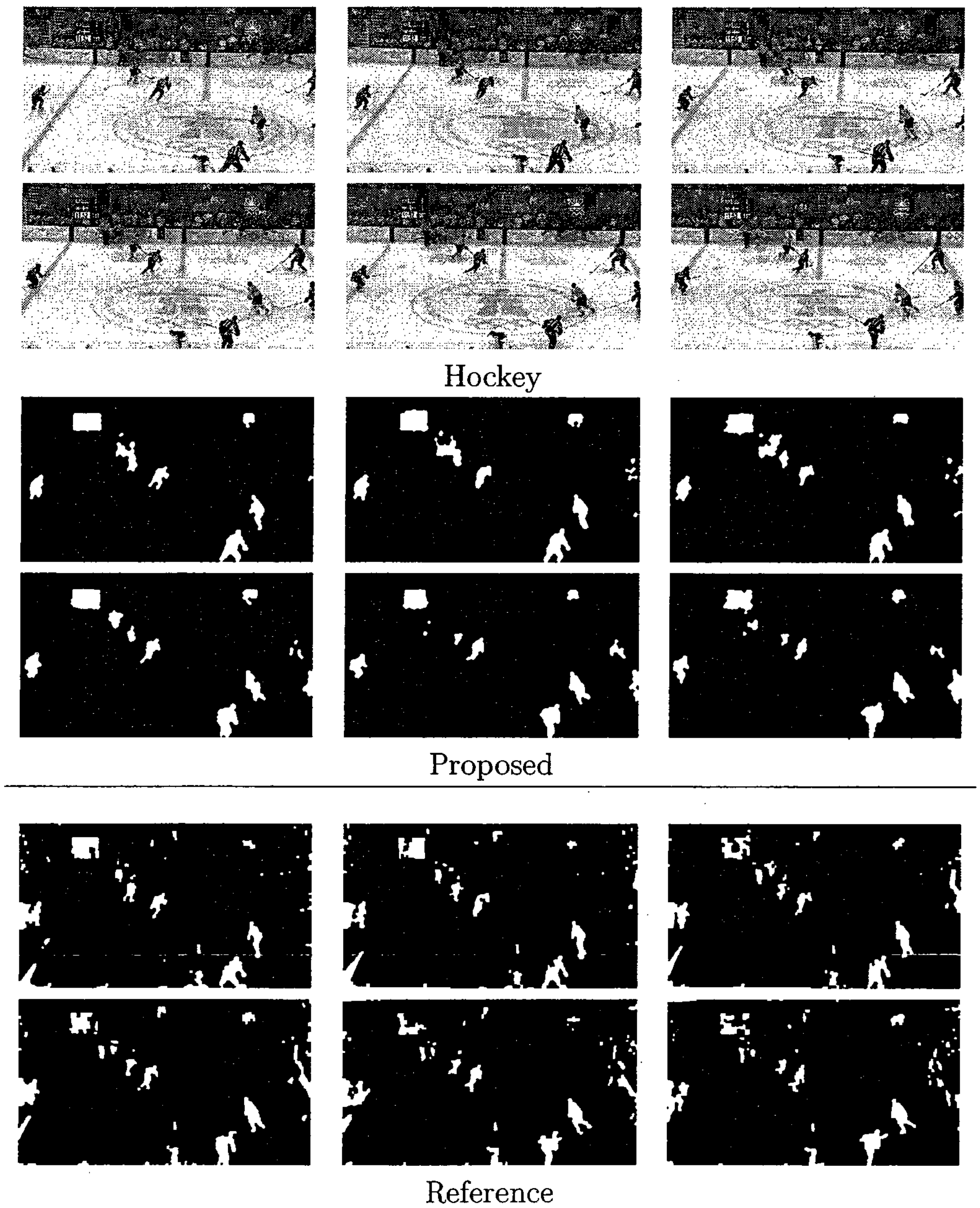


Figure 3.30: Visual comparison of the segmentation output for for frames $l = 52$ to 62 of the *Hockey* sequence between the proposed and reference [6] methods.

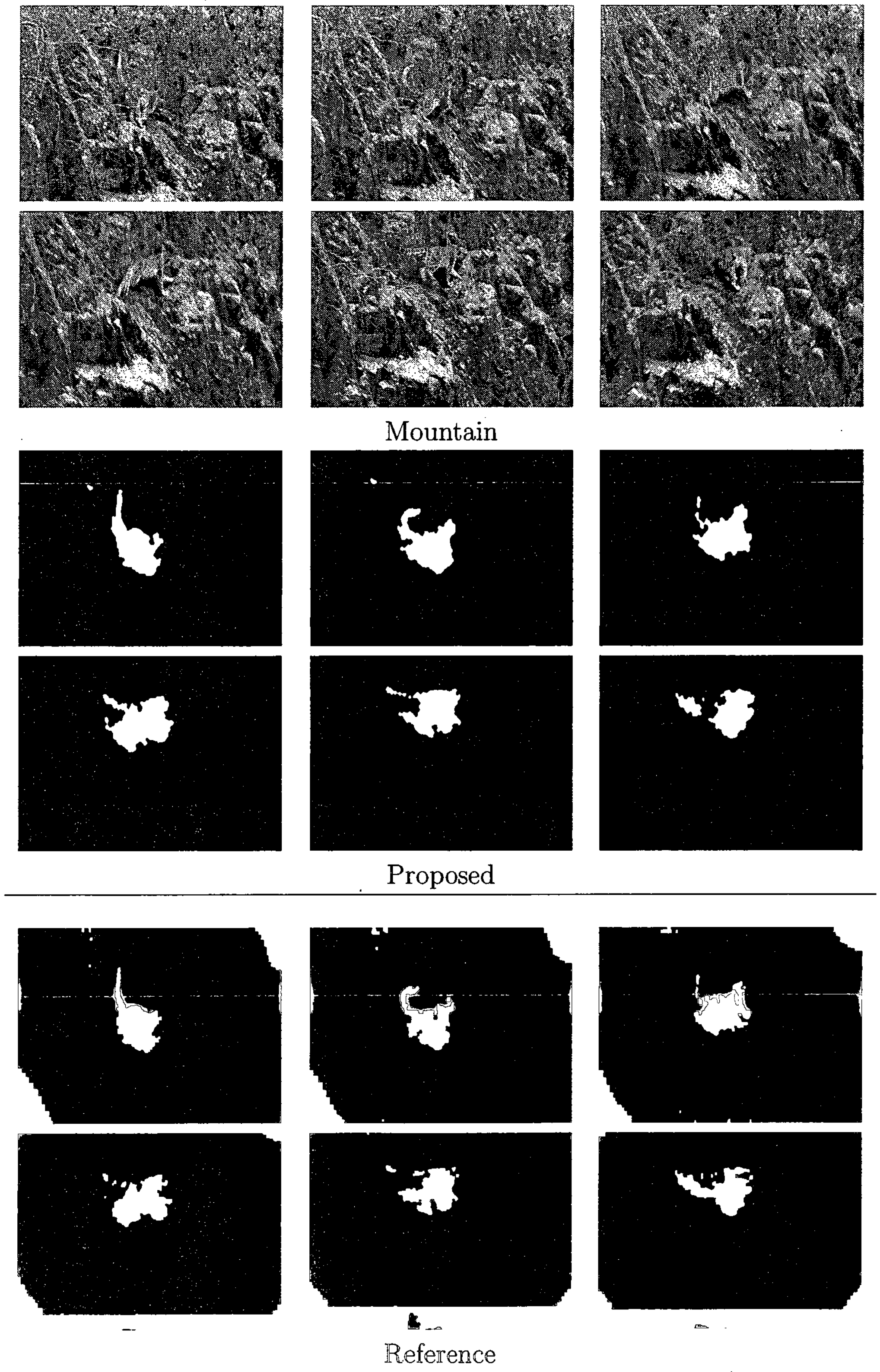
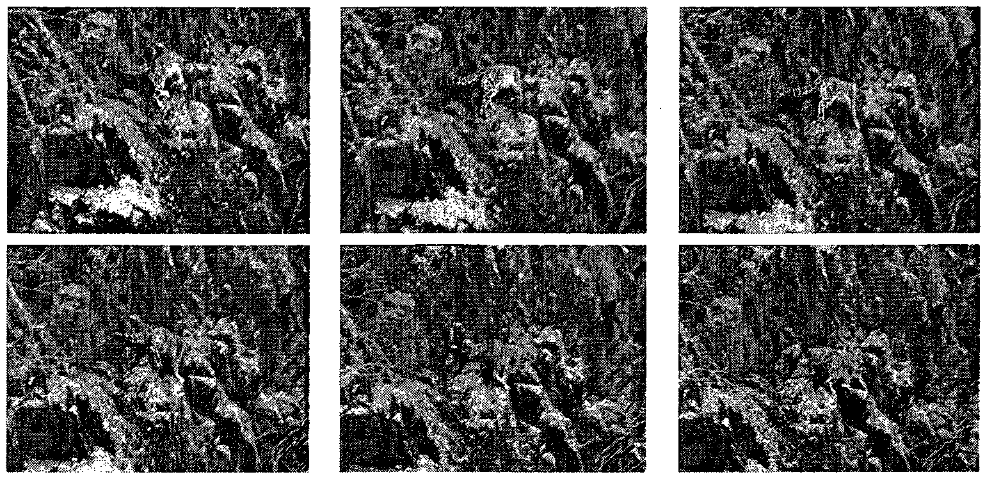
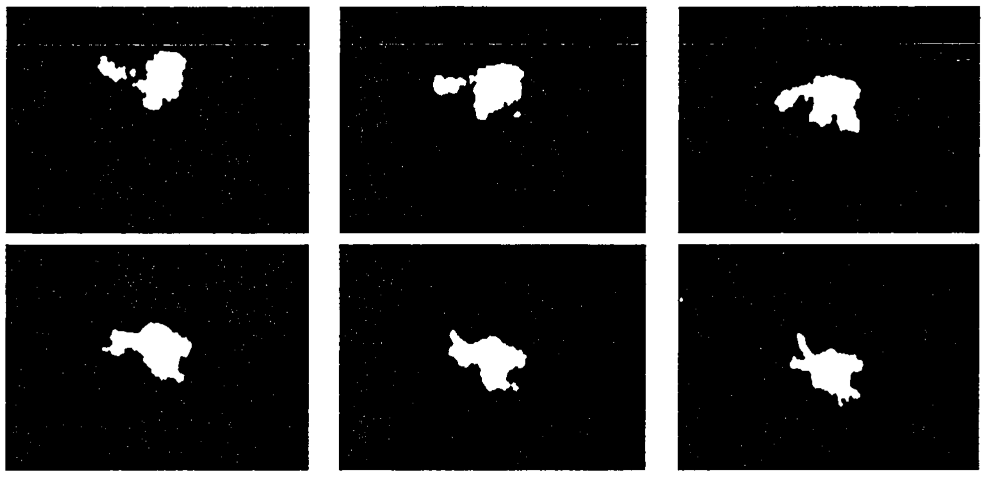


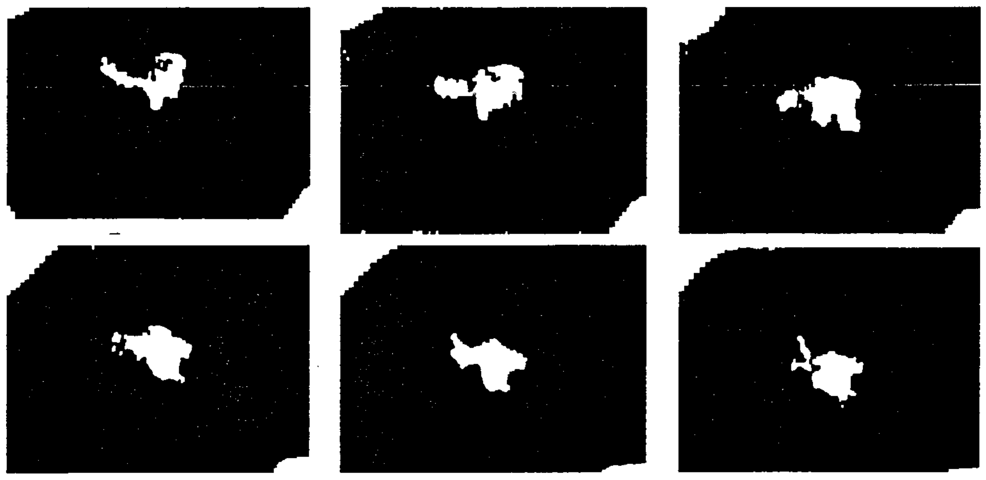
Figure 3.31: Visual comparison of the segmentation output for frames $l = 22$ to 32 of the *Mountain* sequence between the proposed and reference [6] methods.



Mountain



Proposed



Reference

Figure 3.32: Visual comparison of the segmentation output for frames $l = 34$ to 44 of the *Mountain* sequence between the proposed and reference [6] methods.

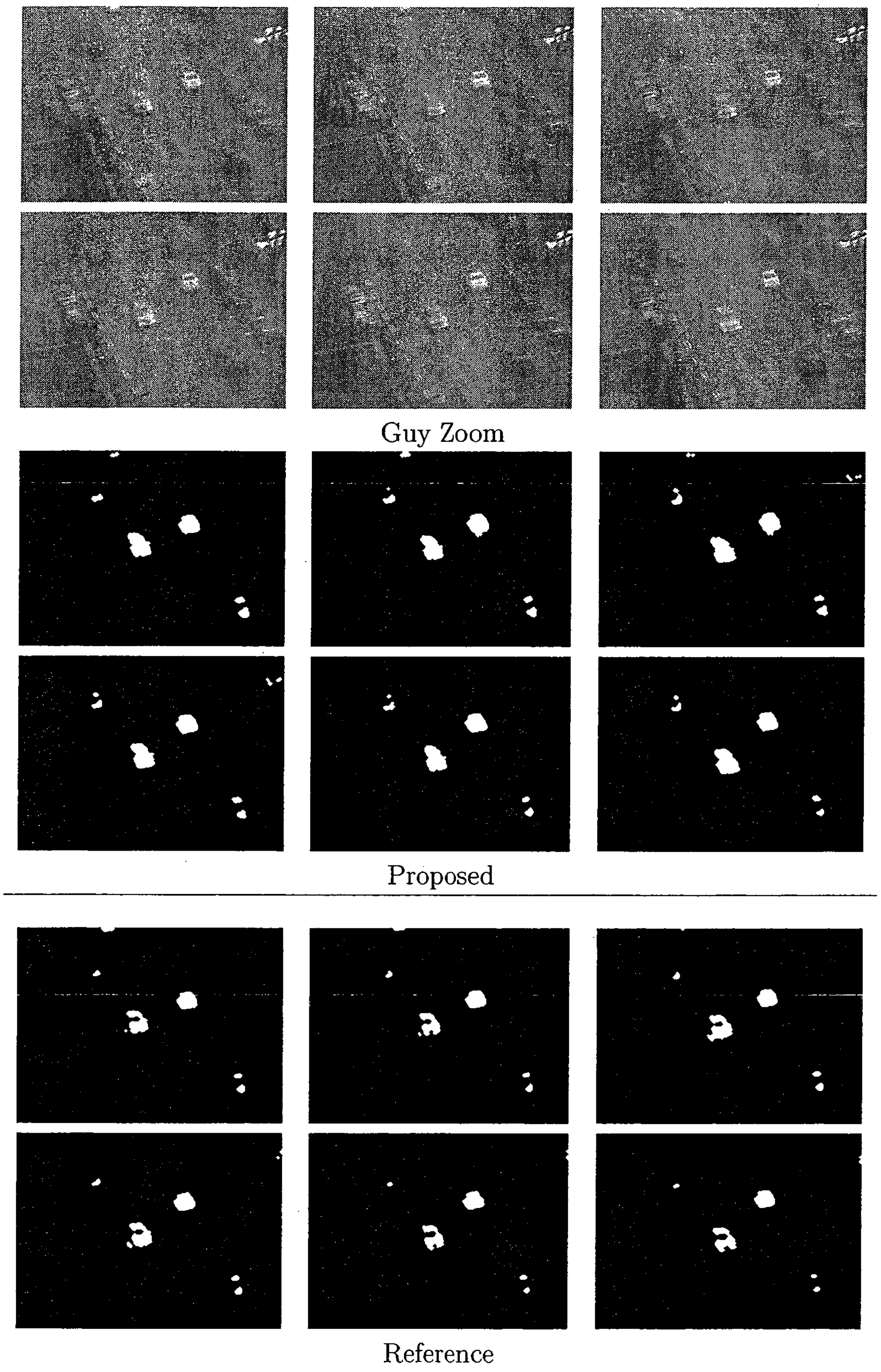


Figure 3.33: Visual comparison of the segmentation output for frames $l = 770$ to 775 of the *Guy Zoom* sequence between the proposed and reference [6] methods.

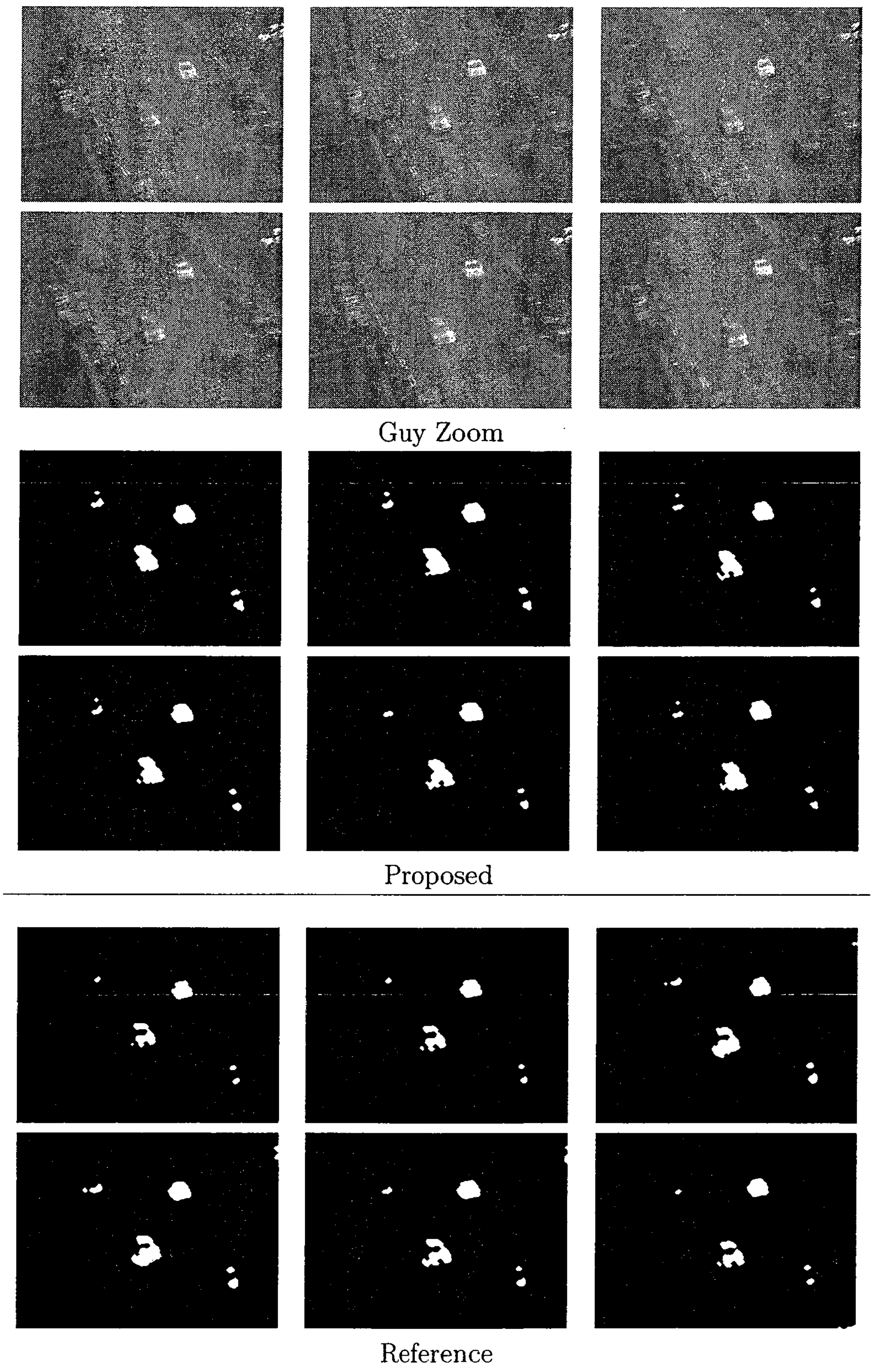


Figure 3.34: Visual comparison of the segmentation output for frames $l = 776$ to 781 of the *Guy Zoom* sequence between the proposed and reference [6] methods.

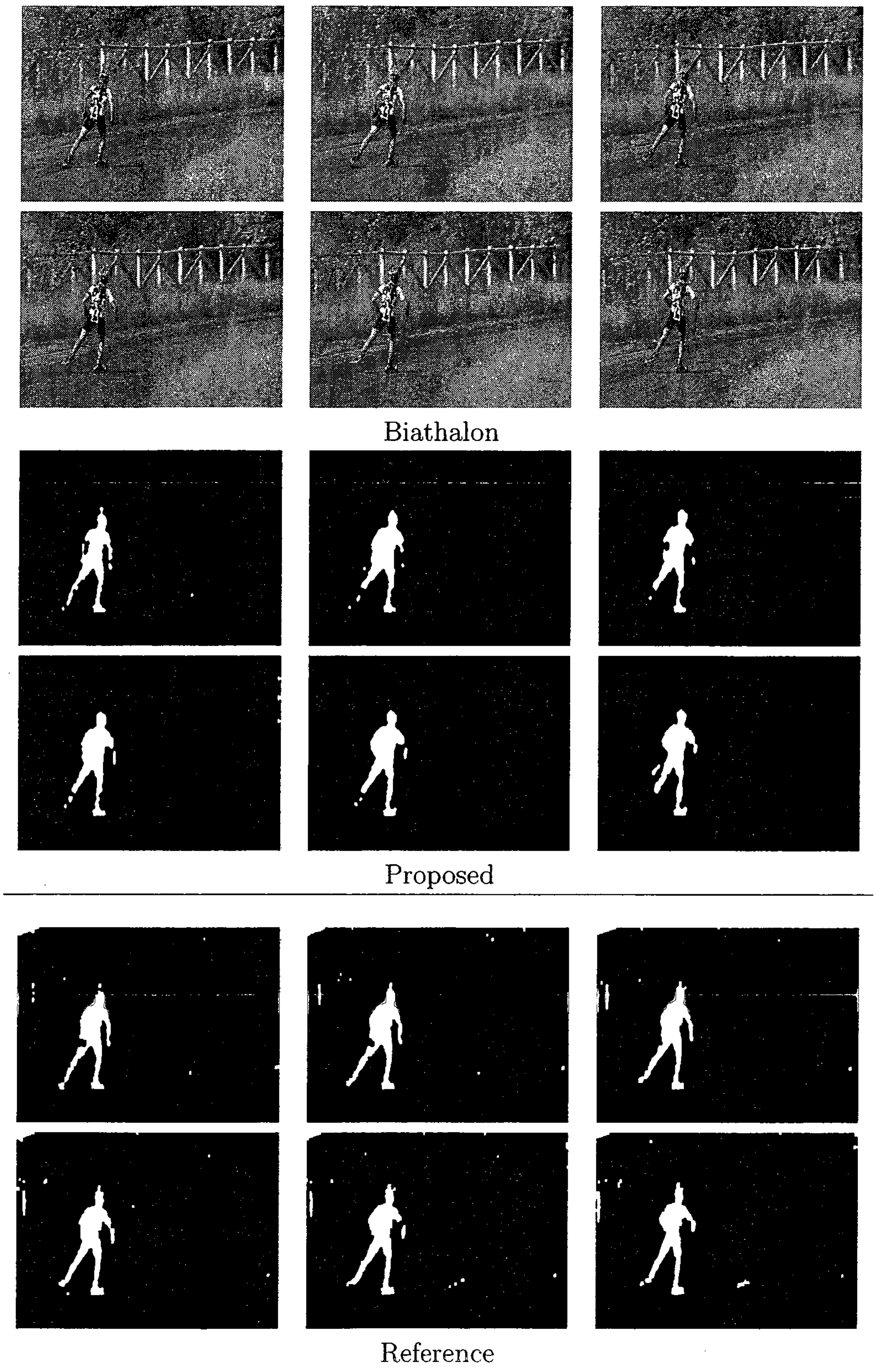


Figure 3.35: Visual comparison of the segmentation output for frames $l = 10$ to 15 of the *Biathalon* sequence between the proposed and reference [6] methods.

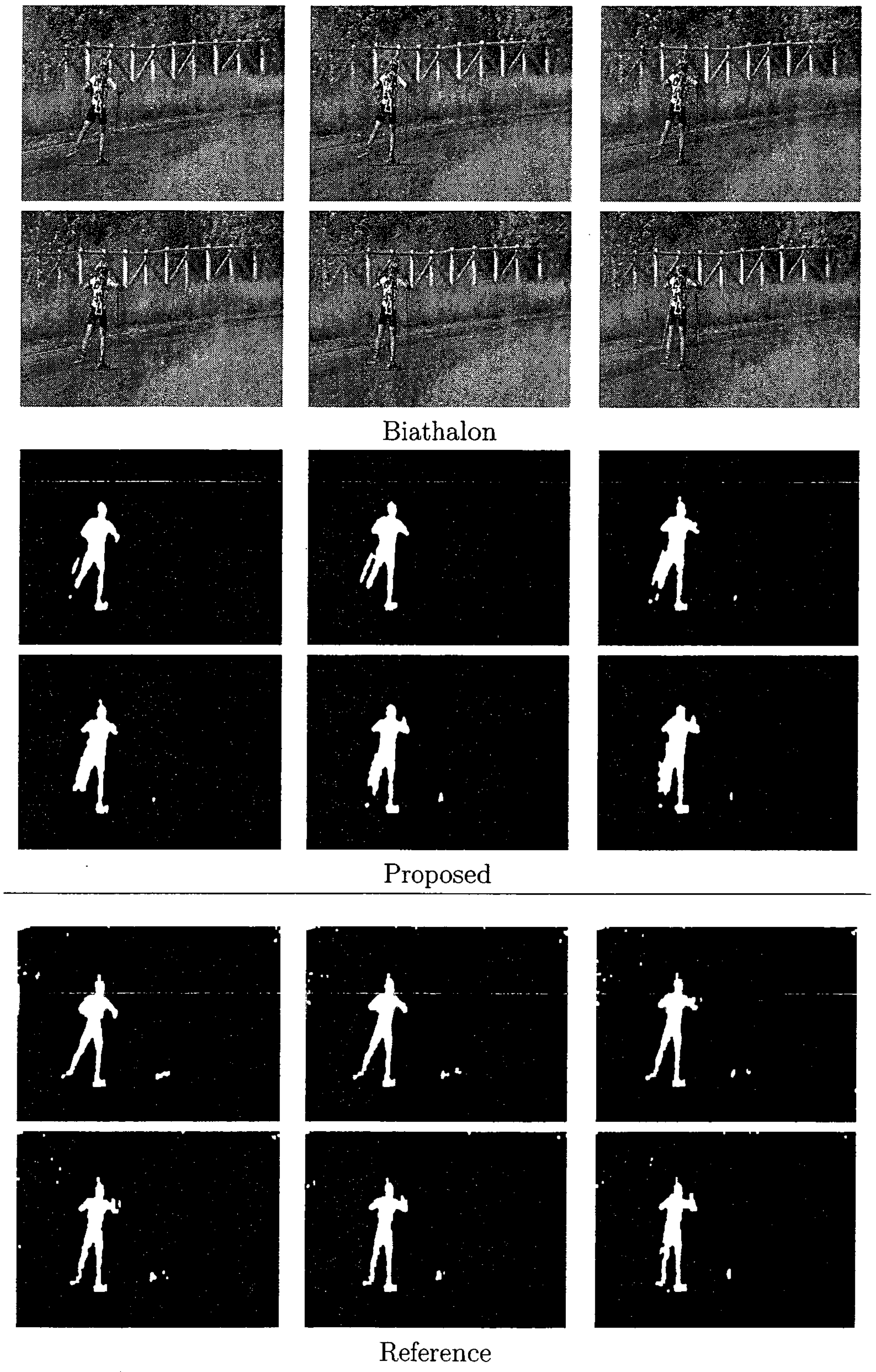


Figure 3.36: Visual comparison of the segmentation output for for frames $l = 16$ to 21 of the *Biathalon* sequence between the proposed and reference [6] methods.

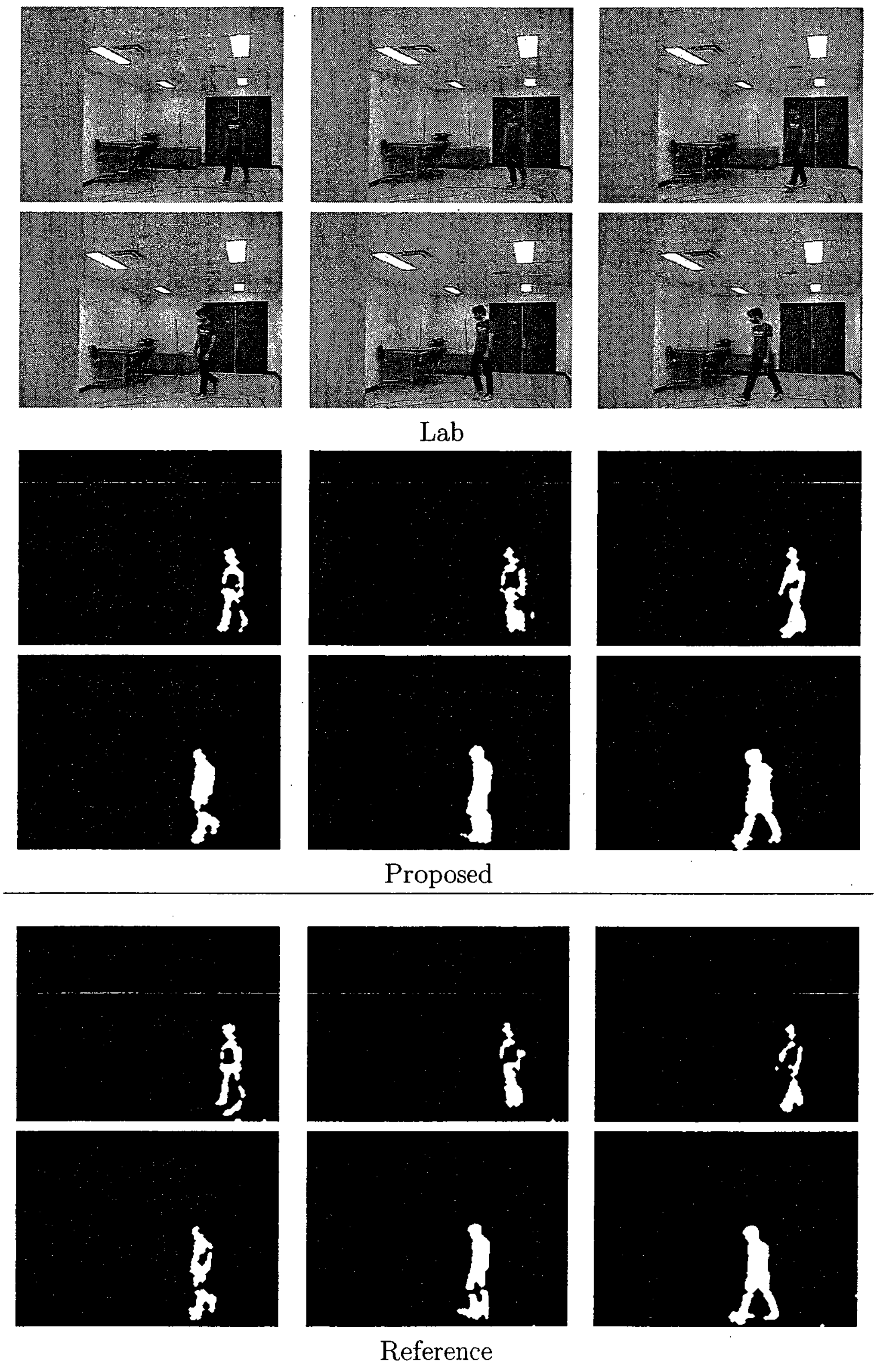


Figure 3.37: Visual comparison of the segmentation output for frames $l = 302$ to 319 of the *Lab* sequence between the proposed and reference [6] methods.

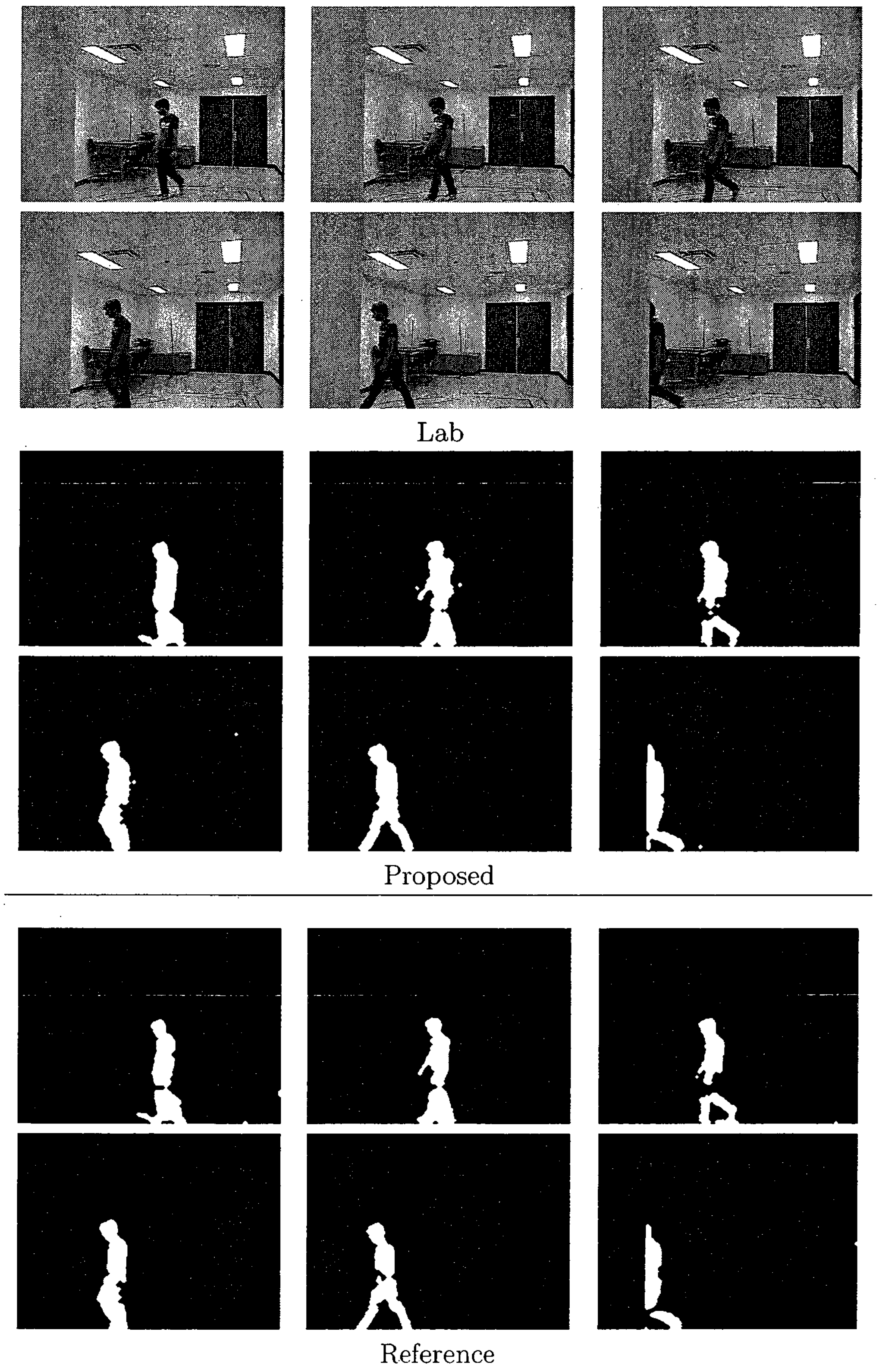


Figure 3.38: Visual comparison of the segmentation output for frames $l = 320$ to 335 of the *Lab* sequence between the proposed and reference [6] methods.

Chapter 4

Object Tracking with Occlusion and Split Handling

4.1 Introduction

Tracking is the process of creating temporal links between objects in consecutive frames. Reliability and computational complexity are two important performance measures for tracking. Reliable trackers maintain for each tracked object a unique identity during its lifetime in the scene. This chapter proposes a novel algorithm for the real-time detection and correction of occlusion and split in object tracking for surveillance applications. The detection and correction of occlusions is referred to in this thesis as handling occlusion. The chapter assumes a feature-based model for tracking and is based on the identification of sudden variations of spatio-temporal features of objects to detect occlusions and splits. The detection is followed by a validation stage that uses past tracking information to prevent false detection of occlusion or split. Special care is taken in case of heavy occlusion, when there is a large superposition of objects. For the detection of splits, in addition to the analysis of spatio-temporal changes in objects features, our algorithm analyzes the temporal behavior of split objects to discriminate between errors in segmentation and real separation of objects, such as in a deposit event. Both objective and subjective experimental results show the ability of the proposed algorithm to detect and correct, both, split and occlusion of objects. The proposed algorithm is suitable in video surveillance applications due to its good performance in multiple, heavy, and total occlusions, its ability to differentiate between real object separation and faulty object split, its handling of simultaneous occlusion and split events, and its low computational complexity. The algorithm was integrated into an online video surveillance system and tested under several conditions with promising results.

4.2 Review of Related Work

Successful tracking of objects is a critical step in any automated video surveillance application. Object tracking can be defined as the process of establishing unique correspondences between objects in a video sequence. Establishing of such correspondences provides necessary information to extract high-level semantics from the video sequence such as events and behaviors of objects.

The tracking process is challenged by incidents of occlusion and split. Occlusions occur either due to real occlusion (i.e., one or more objects mask or overlap regions of other objects in the 2D image plane) or due to limitations of the segmentation algorithm used (e.g., when an object is split into segments). Events of partial or total occlusion are common in real video sequences with multiple objects. Splits occur either because of real split (e.g. an object deposits another object or two or more objects physically move apart or separate) or because of errors in the segmentation algorithm due to illumination changes, shadows and noise.

A good object tracker detects both real and erroneous types of occlusion and split and corrects the tracked object features accordingly. Many tracking algorithms have dealt with these problems [7, 8, 10, 71–78]. In most of the cases, the detection and correction of occlusion and split is dictated by how tracking is performed.

Tracking algorithms can be categorized into *template-based*, *region-based*, and *feature-based*. They can also be categorized into segmentation-based or without prior segmentation. Template-based trackers do not require object segmentation as opposed to region and feature based trackers. In template-based tracking [71–73], the features of tracked templates are learned in the initialization phase of the tracking process. The tracker then searches the frame for these features. Occlusion is detected by the absence of the template features in the frame beyond a certain threshold. Objects in such algorithms are not tracked during occlusion but after object reappearance. While such algorithms work well for tracking of single objects, they fail to robustly track multiple objects during occlusion. Split is not explicitly detected, however, if the object is split, the minimization of the template's feature-comparison function chooses to which portion of the split object the match is made.

Particle filters [79–83] have recently gained popularity in the implementation of template based trackers. They offer good results for the case of single or limited multiple object tracking. However, particle filter-based tracking methods are challenged by the nonlinear characteristics of objects and of the observation model, which requires them to often run at high update rates [82]. Moreover, in many applications, the prior information available for the environment is limited. The complexity of the tracking process increases with multiple objects and multiple cameras. This is especially true in a video surveillance setting with objects frequently entering, leaving and occluding in the scene. With multiple objects, particle filter tracking is hard and the performance highly depends on the ability to account for all statistically significant modes with enough samples [84]. Particle filters are known to handle partial short-duration occlusion, but sometimes lose some objects when multiple objects are involved or when the occlusion is heavy or total as the features of the objects are increasingly missing from the scene. When objects are lost due to an occlusion, updating the models can pose a problem for the tracker later when occlusion ends [83].

A layer-based tracking algorithm is presented in [85]. It divides the frame into multiple depth-ordered layers. Occlusions are detected by defining the pixel-wise layer visibility used to make the system aware of the pixels in the outer most foreground layer that are occluding pixels in one or more foreground or background layers. Handling of occlusion is through prevention of object update during occlusion. The main drawback of this method is associated to the high computational cost of computing and maintaining layer information.

Region-based algorithms [8, 74] divide objects into spatial or motion consistent regions. Region descriptors are projected into and corrected in the next frame. Occlusion is detected when a set of pixels is found to belong to two or more regions and is corrected by separating the occluding objects. Split is detected when a disconnected region is found with region descriptors matching a previously connected object region. Split objects are merged to the object with minimum region descriptor difference [74] or closest in terms of distance [8].

Feature-based tracking algorithms [7, 10, 75–77] model objects using a set of extracted features such as object shape, contour, motion, color or histogram. Tracking is performed by solving the object matching problem. Occlusion in such algorithms is detected as a sudden change in object features and is corrected by temporal filtering. Kalman filtering is often used for temporal object prediction [76]. Feature-based tracking algorithms are sometimes confused by object disappearance in case of total occlusion. The method in [10] identifies and handles long-lived occlusions by creating a group that contains the information from the occluded objects. Once the occlusion is finished the algorithm identifies a split and correctly label the separated objects.

In this chapter, we propose a novel algorithm for the real-time detection and correction of occlusion and split in object tracking for surveillance applications. We assume a feature-based model for tracking and analyze the spatio-temporal features of objects to detect sudden variations indicating a possible occlusion or split. Detection is followed by a validation stage using past tracking information to prevent false detection of occlusion or split. In the detection and correction of occlusions we take special care in treating heavy occlusion, when there is a large superposition of objects. In this case the system avoids updating the video objects features with unreliable (e.g. shape and motion) information. Occlusion is corrected by separating occluded objects. For split detection, in addition to the analysis of spatio-temporal changes in objects features, our algorithm analyzes the temporal behavior of split objects to discriminate between errors in segmentation and real separation of objects, such as in a deposit event. Split is corrected by physically merging the split objects.

The remainder of the chapter is organized as follows. Section 4.3 gives an overview of the proposed algorithm. Section 4.3.1 introduces the proposed scheme for occlusion detection and handling. Section 4.3.2 introduces the proposed scheme for split detection and correction. Section 4.4 presents objective and visual simulation results, and Section 4.5 summarizes the chapter.

4.3 Proposed object tracking

The terms *Image Object* and *Video Object* are used extensively throughout this section. *Image Object* (IO) identifies a closed contour in the current frame F_n. The main characteristic of this entity is that it does not include temporal information of the object. An image object lives only in the current frame. *Video Object* (VO) refers to a temporally consistent object with temporal information, such as motion, trajectory, or visibility. A video object is updated at the end of the tracking process with the information of the matched image object. A *Video Object* lives throughout the video sequence. Two other concepts frequently used in this section are those of

Stable Objects and *Active Objects*. A *Stable Object* refers to a *Video Object* that has been in the scene enough time (normally more than one second for people surveillance applications) to have temporally stable features. This means that features like size, shape and position have little variation (less than $\pm10\%$ from frame to frame) in time. An *Active Object* is one that is currently in the scene. An object becomes inactive when it exits the scene, or it becomes completely occluded by another object.

The proposed algorithm receives as input a list of image objects $\{\mathrm{IO}_i\}_n$ detected in the current frame F_n and described by closed contours. The image object is labeled by $i \in \{1, 2, .., N_n\}$ with N_n being the number of image objects in frame F_n. It also receives as input a set of video objects $\{\mathrm{VO}_j\}_{n-1}$ from the previous frame F_{n-1}. The video objects are labeled by $j \in \{1, 2, ..N_{n-1}\}$ with N_{n-1} denoting the total number of video objects detected and still active in the previous frame F_{n-1}. The output of the algorithm is the updated set of video objects $\{\mathrm{VO}_j\}_n$ after matching and correction of occlusion and split. To avoid simultaneous split and occlusion errors, at the end of the occlusion and split correction steps, we validate all matches between image and video objects. The validation stage consists of evaluating the confidence level for each match based on a confidence threshold. Only the matches that have a large confidence (larger than 80%) are considered good matches and are kept for the next stage. The matches that do not classify as good matches (confidence below 80%) are removed and we repeat the process of occlusion and split detection. Two or three iterations are normally sufficient to solve all the problems of simultaneous split and occlusion.

The main steps of the proposed algorithm are (see Fig. 4.1):

1. Match objects in $\{\mathrm{IO}_i\}_n$ to objects in $\{\mathrm{VO}_j\}_{n-1}$ based on feature similarity using [7], where image and video objects are characterized by a set of simple features (e.g. size, shape, position, and, for the video object, trajectory and motion). The previous frame is represented in Fig. 4.1 by a one frame delay

(i.e., z^{-1}).

2. Detect and fix occlusions and update $\{IO_i\}_n$ (see Section 4.3.1).

3. Detect and fix split and update $\{IO_i\}_n$ (see Section 4.3.2).

4. Validate correspondences between objects in $\{IO_i\}_n$ and $\{VO_j\}_{n-1}$.

5. If validation fails, go to step 1 (possible simultaneous split/occlusion).

6. Construct the new $\{VO_j\}_n$ using $\{VO_j\}_{n-1}$, the corrected $\{IO_i\}_n$ and the validated matches.

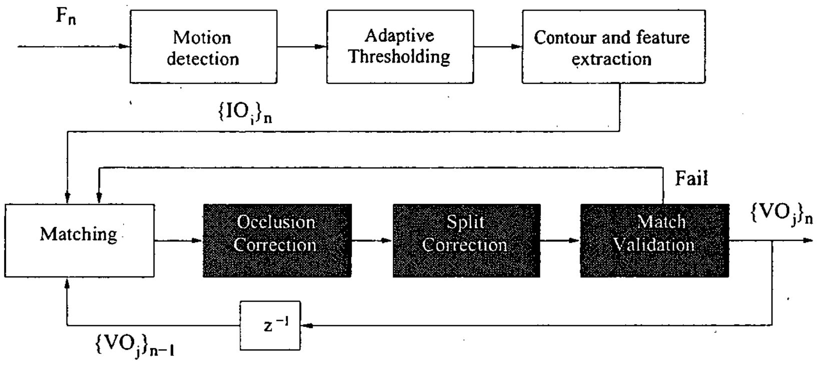


Figure 4.1: Block diagram of tracking algorithm [7], where the proposed occlusion and split handling are integrated to test their effectiveness.

In the matching step a video object is compared to near-by image objects and a confidence level is assigned to each comparison (or correspondence). An image object is near a video object if it is located inside a spatial neighborhood whose size and center are determined using the size, center and previous motion of the video object. The image object with the best correspondence wins the match to that video object. While building the correspondence each feature of the image and video objects contributes some votes to the correspondence, increasing or decreasing its confidence level.

4.3.1 Occlusions detection and correction

Occlusion detection

The detection of occluded objects is based on the detection of significant deviations in object features from the previous state of the tracking algorithm. The main idea is to find which object in the current frame has deviated enough from its previous features to trigger the occlusion detection process. This detection process will take into account the previous tracking state as well as the information from the current frame. To accomplish this, a set of occlusion detection rules are defined to find video objects VO_l and VO_k that are active in the video object list $\{\mathrm{VO}_i\}_{n-1}$ of the previous frame and that have occluded to form the image object IO_{occ} in the image object list $\{\mathrm{IO}_i\}_n$ for the current frame. Every image object IO_i in the current frame matched to a stable video object VO_l is considered a candidate for occlusion and is checked for compliance with all the presented rules ('and' operation).

Rule 1 The video object VO_l is stable, active and matched to an image object IO_i in the current frame F_n. This video object VO_l belongs to the set $\{\mathrm{VO}_l\}_{\mathrm{pocc}}$ of possible occlusions if:

$$\begin{cases} \mathrm{VO}_l \in \{\mathrm{VO}_j\}_{n-1} & \wedge \\ a(\mathrm{VO}_l) > \mathrm{T}_{\mathrm{age}} & \wedge \\ \exists i \in \{1, \ldots, N_n\} \mid \mathit{match}(\mathrm{VO}_l, \mathrm{IO}_i) > \mathrm{T}_{\mathrm{corr}} & \end{cases}, \tag{4.1}$$

where $a(\cdot)$ returns the age (number of frames) of the objects and $\mathrm{T}_{\mathrm{age}}$ is an age threshold defining the minimum age an object stays in the scene to be considered stable. The matching function $\mathit{match}(\cdot,\cdot)$ returns the confidence level of the match between the considered video and image objects. This confidence level is then compared to the match- or correspondence-confidence threshold $\mathrm{T}_{\mathrm{corr}}$ to decide if the match is accepted. All objects in this list are checked with rules

2–5 to verify occlusions.

Rule 2 There is a significant outward change at one of the sides of the Minimum Bounding Box (MBB) of the matched video object VO_l. In order to verify if there is a significant outward change at a given side of the MBB, we use a direction function $r(D)$ which returns a logical value indicating a change at the corresponding side.

$$r(D) = \begin{cases} |m_p(D) - m_c(D)| & \geq \mathrm{T_{chg}} \quad \wedge \\ o(D)\,(m_c(D) - m_p(D)) & \geq 0 \end{cases}, \tag{4.2}$$

where $D \in \{\mathit{Left}, \mathit{Right}, \mathit{Up}, \mathit{Down}\}$ is a side in the object's MBB. The motions $m_p = (m_{x_p}, m_{y_p})$ and $m_c = (m_{x_c}, m_{y_c})$ are, respectively, the previous motion of the video object VO_l, and its estimated motion at the current frame. The function $o(D)$ returns 1 or -1 depending on which side is verified for an outward change (possible occlusion):

$$o(D) = \begin{cases} 1 & \mathit{Right} \vee \mathit{Down} \\ -1 & \mathit{Left} \vee \mathit{Up} \end{cases}, \tag{4.3}$$

This $o(\cdot)$ function is introduced to take into account the fact that an outward change in the direction of the axis (*Right* or *Down*) is positive, while an outward change in the opposite direction (*Left* or *Up*) is negative. Motion m_c(D) is the motion of the corresponding side of the object's MBB. The threshold $\mathrm{T_{chg}}$ defines the amount of change in the motion of a side that we consider a deviation from a normal behavior.

Rule 3 There is a unmatched object $\mathrm{VO}_k \in \{\mathrm{VO}_j\}_{n-1}$ close to a side D of the object VO_l under consideration (the matched object) where a change has been detected, i.e.

$$\exists \mathrm{VO}_k \mid \begin{cases} match\,(\mathrm{VO}_k, \mathrm{IO}_i\,) & < & \mathrm{T_{corr}} \quad \forall i \in \{1, \ldots, N_n\} & \wedge \\ d\,(\,\mathrm{VO}_k, \dot{\mathrm{VO}}_l\,) & < & \mathrm{T_{dist}} & \wedge \\ p\,(\,\mathrm{VO}_l, \mathrm{VO}_k\,) & = & D & \wedge \\ r\,(\,D\,) & = & \text{true} & \end{cases}, \quad (4.4)$$

where the function $d\,(\,\cdot,\cdot\,)$ refers to the Euclidean distance between two objects and $\mathrm{T_{dist}}$ is a distance threshold used to decide when two given objects are close to each other. The function $p\,(\,\mathrm{VO}_l, \mathrm{VO}_k\,)$ returns the relative position of VO_k with respect to VO_l.

Rule 4 The area A of the image object resulting from the occlusion IO_{occ} is larger than the area of the smaller of the two objects (VO_k and VO_l).

$$A\,(\,\mathrm{IO}_{occ}\,) > \min(A\,(\,\mathrm{VO}_k\,)\,, A\,(\,\mathrm{VO}_l\,)). \quad (4.5)$$

Rule 5 The unmatched object VO_k has also experienced, like the matched object VO_l (see Rule 2), a significant change in the corresponding side of the MBB if matched to the same image object IO_{occ}.

$$r\left(\,\bar{D}\,\right) \;=\; \text{true for } \mathrm{VO}_k \;\text{ if }\; r\,(\,D\,) \;=\; \text{true for } \mathrm{VO}_l\;, \quad (4.6)$$

where $\bar{D}$ refers to the side of the MBB directly opposite to D.

Declaration of occlusion

Once we have analyzed a candidate occluded object and decided that it is occluded, we declare an occlusion event containing the identification of the involved video objects, the identification of the image object and the estimated positions (only the MBBs) of the occluded video objects in the current frame as

$$\mathrm{Occ}_i = \{\mathrm{IO}_{occ}, \mathrm{VO}_l, \mathrm{VO}_k, \widetilde{\mathrm{MBB}}(l,n), \widetilde{\mathrm{MBB}}(k,n)\}, \tag{4.7}$$

where IO_{occ} is the image object (in the current frame) that contains the two occluded video objects VO_l and VO_k. The estimated MBBs, $\widetilde{\mathrm{MBB}}(l,n)$ and $\widetilde{\mathrm{MBB}}(k,n)$, respectively designate the estimated Minimum Bounding Boxes of video objects VO_l and VO_k in the current frame F_n.

We distinguish between two different cases for the estimation of the position of the occluded video objects VO_l and VO_k in the current frame:

1. The occlusion produces a small overlapping of the involved objects. In this case, the occluded objects preserve most of their shape, making their extracted features, including motion, reliable. We estimate the new positions by using the last known motion of each object, i.e. the motion in the previous frame F_{n-1},

$$\begin{aligned} \widetilde{\mathrm{MBB}}(k,n) &= \mathrm{MBB}(k,n-1) + m_p(\mathrm{VO}_k), \\ \widetilde{\mathrm{MBB}}(l,n) &= \mathrm{MBB}(l,n-1) + m_p(\mathrm{VO}_l), \end{aligned} \tag{4.8}$$

   where $m_p(\mathrm{VO}_{k(l)})$ is the motion of the video object $\mathrm{VO}_{k(l)}$ in the previous frame F_{n-1}. The "+" operation in this context means that the MBB of the video object in the previous frame is moved based on the motion of the object in the previous frame.

2. The occlusion involves significant parts of the objects (heavy occlusion). In this case, when the overlapping of objects is very high, it is possible to temporarily "lose" one of the objects. If the unmatched object VO_k is previously "lost", its features, especially its motion, are unreliable. Using those features to estimate the new position on the object could result in an inaccurate estimation. In

this case we make use of the average motion of the object *before* the occlusion started. This way we avoid using inaccurate motion in the prediction stage. The estimation of the new positions in case of heavy occlusion is then

$$\begin{aligned} \widetilde{\mathrm{MBB}}(k,n) &= \mathrm{MBB}(k,L) + (n-L)\bar{m}_L(\mathrm{VO}_k), \\ \widetilde{\mathrm{MBB}}(l,n) &= \mathrm{MBB}(l,n-1) + m_p(\mathrm{VO}_l), \end{aligned} \tag{4.9}$$

where L is the last frame where the object was not occluded ($n-1$ if the object is not "lost"). The motion $\bar{m}_L$ refers to the average motion of the video object in frame F_L and is multiplied by the time difference to account for the time the object is "lost". Once the heavy occlusion is finished, objects recover their normal shape. The motion is then re-estimated based on the position of the objects before the occlusion started and the new position after the occlusion.

At the end of the occlusion detection process we have a list $\mathrm{OL}_n = \{\mathrm{Occ}_i\}_n \quad i \in \{1,\ldots,O_n\}$ of occlusion events each containing the necessary information to handle the occlusion. O_n is the number of occlusion events in the current frame. It is important to note that there is an occlusion event for each unmatched video object VO_k in the previous frame that is detected as occluded in the current frame. The number of occlusion events O_n may then be different from the number of image objects in the current frame F_n containing occluded video objects. These two numbers differ in case of multiple occlusions as we explain in the following section.

Multiple occlusions

The proposed set of rules (1–5) return a list of occlusion events related to video objects from the previous frame that are occluded into a single image object in the current frame. In the simplest case, two video objects are occluded into a single image object. This case is easily handled by separating the image object into two objects that may be matched to the two occluded video objects. It is also possible, however, to have a case of multiple occlusions.

Multiple occlusions happen when more than two video objects are occluded into a single image object. In this case we detect several occlusion events each including the common image object IO_{occ}, the video object that is matched to it VO_l, and one of the unmatched but occluded video objects VO_k. The occlusion list OL_n contains one occlusion event for each unmatched video object that is detected to be occluded. An example of multiple occlusion is depicted in Fig. 4.2. In this figure three video objects from the previous frame (VO_1, VO_2 and VO_3) are occluded to form the image object IO_1 in the current frame.

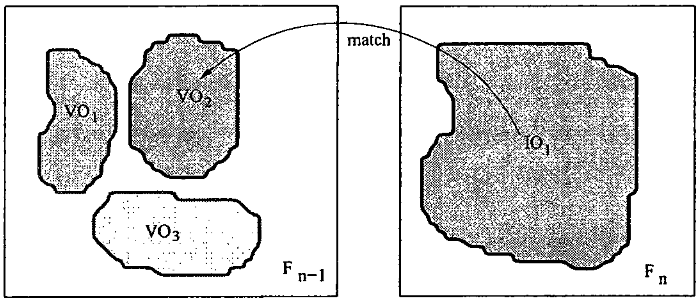


Figure 4.2: Multiple occlusions example.

The occlusion list OL_n corresponding to this case will be composed of two occlusion events:

$$\mathrm{Occ}_1 = \{\mathrm{IO}_1, \mathrm{VO}_2, \mathrm{VO}_1\} \qquad \mathrm{Occ}_2 = \{\mathrm{IO}_1, \mathrm{VO}_2, \mathrm{VO}_3\}\ , \tag{4.10}$$

where the common video object VO_2 is the one matched to IO_1.

In order to handle the multiple occlusion cases, we modify the occlusion events Occ_i in such a way that the matched video object is not repeated in the events of the occlusion list. We transform the multiple occlusion events list which always involves an image object and the matched video object into a list of occlusions between two video objects. This transformation facilitates handling of the occlusion events. For example, in Fig. 4.2, where the matched video object VO_2 is present in both occlusion events, we need to modify the events to ensure that the matched object VO_2 is only present in one of the events. The events of Eq. (4.10) are then modified to:

$$\widetilde{\mathrm{Occ}}_1 = \{\mathrm{IO}_1, \mathrm{VO}_2, \mathrm{VO}_1 \bigcup \mathrm{VO}_3\} \qquad \widetilde{\mathrm{Occ}}_2 = \{\mathrm{IO}_1, \mathrm{VO}_1, \mathrm{VO}_3\} \ , \tag{4.11}$$

The estimated MBB for the second video object (VO_1) in the first event (Occ_1) accumulates the estimated MBBs of all the other objects involved in the occlusion. In general, for a given occlusion event Occ_j, the MBB of the second object in the event contains the MBBs of all the other video objects appearing in the list that share the same image object. Let J and J_M be, respectively, the first and last index in the occlusion list OL_n of the occlusion events involving image object IO_{occ}, then the MBBs in the occlusion events are transformed as:

$$\widetilde{\mathrm{Occ}}_j.\mathrm{MBB}(1) = \mathrm{Occ}_{j-1}.\mathrm{MBB}(2) \quad \forall j \in \{J+1, \ldots, J_M\}, \tag{4.12}$$

$$\widetilde{\mathrm{Occ}}_j.\mathrm{MBB}(2) = \bigcup_{i=j+1}^{J_M} \mathrm{Occ}_i.\mathrm{MBB}(2) \quad \forall j \in \{J, \ldots, J_M - 1\}, \tag{4.13}$$

where $\mathrm{Occ}_i.\mathrm{MBB}(2)$ is the MBB of the second video object involved in the occlusion Occ_i. The purpose of following this path will become clear in the next section, but the main idea behind modifying the occlusion events is to reduce the multiple occlusions to a series of two objects occlusions.

Occlusion correction

To correct detected occlusions, we scan the modified occlusion list OL_n of the current frame and correct one occlusion at a time. Since the list was modified in such a way that each occlusion event involves just two video objects and one image object, correcting an occlusion event reduces to separating the image object into two objects based on the estimated positions of the video objects involved in the occlusion event.

The correction strategy we propose is to extract one video object in each correction step. The extracted object is the one matched to the common image object. The steps of the occlusion correction process are:

1. We pick an occlusion event Occ_i from the occlusion list OL_n. This event involves an original image object IO_{occ}, the video object VO_l matched to it and an occluded video object VO_k containing the accumulated MBBs of all the other occluded objects in case of multiple occlusions.

2. We extract from the image object IO_{occ} in the current frame the part that corresponds to the matched video object VO_l and the rest of the image object becomes a new image object $\mathrm{IO}_{occ,2}$.

3. This new image object $\mathrm{IO}_{occ,2}$ is then assumed to be matched to VO_k (the unmatched object in the first occlusion event).

4. If the next occlusion event Occ_{i+1} involves the same image object IO_{occ}, we update the event with the new image object $\mathrm{IO}_{occ,2}$ and handle the occlusion Occ_{i+1}. The occlusion event Occ_{i+1} involves the new image object $\mathrm{IO}_{occ,2}$, the second object from the previous occlusion event VO_k (now matched to this image object) and an object that represents all the remaining video objects in the multiple occlusion. This is achieved by going to step 2

5. When the next occlusion event involves a new image object, we restart the process from step 1.

6. The process finishes when there is no more occlusion events in the list.

In Fig. 4.3 we depict the process of correcting the occlusion in the example shown in Fig. 4.2. In this figure the first step is to extract VO_2, which is the matched object. The resulting image object contains the information for VO_1 and VO_3. In the second step we extract VO_1 and VO_3, resulting in three image objects that are then matched to the corresponding video objects. In Fig. 4.3, the extracted image objects are depicted with dashed lines while solid lines represent the MBBs.

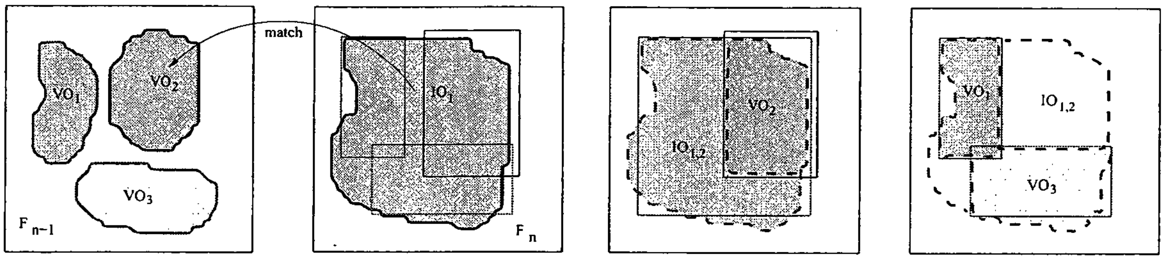


Figure 4.3: Correction of the occlusion scenario in Fig. 4.2

At the end of the occlusion correction stage, we have as many new image objects as video objects from the previous frame that were found to be occluded (and without a match in the current frame). Each video object that is matched have a unique correspondence with an image object and vice-versa.

4.3.2 Split detection and correction

Split detection

The proposed algorithm differentiates between real splits (e.g., due to object deposit) or splits due to segmentation errors (e.g., when one object is fragmented). The detection of split objects is based on a set of rules (1–5) that find an image object IO_l that have split from the image object IO_k which is matched to video object VO_k.

Rule 1 Every unmatched image object is considered as a possible split object. A list $\{\mathrm{IO}_l\}_{\mathrm{pspt}}$ of possible split image objects is build with all unmatched objects

$$\{\mathrm{IO}_l\}_{\mathrm{pspt}} = \left\{ \mathit{match}(\mathrm{IO}_l, \mathrm{VO}_i) < \mathrm{T}_{\mathrm{corr}} \quad \forall i \in \{1, \ldots, N_{n-1}\} \right\} \tag{4.14}$$

Rule 2 There is a matched image object IO_k close to the unmatched image object

IO_l.

$$\exists\, \mathrm{IO}_k \quad | \quad \begin{cases} d\,(\mathrm{IO}_k, \mathrm{IO}_l\,) < \mathrm{T}_{\text{dist}} & \wedge \\ \exists\, k \in \{1, \ldots, N_{n-1}\}\,|\;\; \mathit{match}\,(\,\mathrm{IO}_k, \mathrm{VO}_k\,) \;\; > \;\; \mathrm{T}_{\text{corr}} & \end{cases} \tag{4.15}$$

Rule 3 If the MBBs of the two image objects IO_l and IO_k overlap, we check if there is an overlapping between the unmatched object IO_l and the motion-compensated matched video object in the current frame $\widetilde{\mathrm{MBB}}(k,n)$ as defined in Eq. 4.8.

$$\mathrm{IO}_l \cap \mathrm{IO}_k \neq \emptyset \quad \wedge \quad \mathrm{IO}_l \cap \widetilde{\mathrm{MBB}}(k,n) \neq \emptyset, \tag{4.16}$$

where $\widetilde{\mathrm{MBB}}(k,n)$ is the estimated MBB of the matched video object VO_k in the current frame.

Rule 4 If the MBBs of the two image objects IO_l and IO_k do not overlap, we check if there is a significant inward change at one of the sides of the MBB of image object IO_l

$$\mathrm{IO}_l \cap \mathrm{IO}_k = \emptyset \quad \wedge \begin{cases} p\,(\,\mathrm{IO}_k, \mathrm{IO}_l\,) & = \quad D \quad \wedge \\ r\,(\,D\,) = \;\text{true} & \text{for} \quad \mathrm{IO}_l \end{cases}, \tag{4.17}$$

where

$$r\,(\,D\,) \;\; = \begin{cases} |\, m_p(D) - m_c(D)\,| & \geq \;\mathrm{T}_{\text{chg}} \quad \wedge \\ o\,(\,D\,)\,(m_c(D) - m_p(D)) & \leq \; 0 \end{cases}, \tag{4.18}$$

Rule 5 If the split event is consistent over a time period, (same position, similar area) and the detected image objects are separating. the split event is confirmed as a real separation. In this case, a new video object is created and matched to the unmatched image object IO_l.

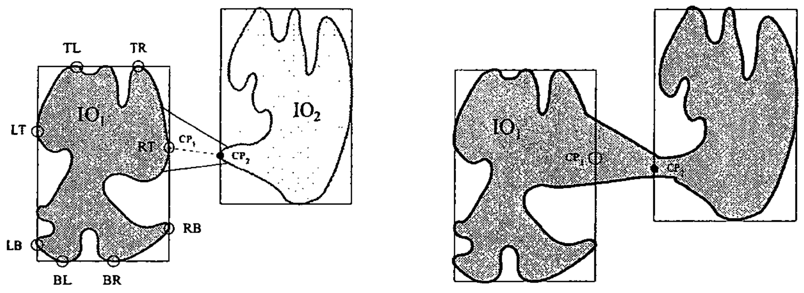
Split declaration

When rules 1, 2 and either 3 or 4 are satisfied, we declare the unmatched image object as part of the matched one that is fragmented by the segmentation method. If rule 5 is verified then the split event is changed to a real separation and the split correction is not carried out.

Once we have verified all the objects in the current frame to detect the fragmented objects, we construct a list $\mathrm{SL} = \{\mathrm{Spl}_i\}_n \quad i \in \{1, \dots, S_n\}$ of split events, where the split event $\mathrm{Spl}_i = \{\mathrm{IO}_l, \mathrm{IO}_k\}$ contains the necessary information to fix the segmentation error. This split list is then passed to the correction algorithm to merge the fragmented objects into one.

Split correction

Once we have detected that two or more image objects in $\{\mathrm{IO}_n\}$ belong to the same video object, we proceed to merge these image objects into one. We process the split event list SL and correct splits by merging two objects each time. In the case of an image objects that is split into several image objects, we make several merges. The process is accomplished by drawing a solid line connecting the closest points of the contours. Let IO_1 and IO_2 denote the objects to merge. To facilitate merging, for each contour (image object) we identify the points that lie on the sides of the bounding box (MBB). We keep and store two points for each side of the MBB, the points closer to the corners of the MBB. Those eight points are labeled: Top-Left, Top-Right, Bottom-Left, Bottom-Right, Left-Top, Left-Bottom, Right-Top, and Right-Bottom. Fig. 4.4 shows an example of correction of split. In Fig. 4.4 (a) we show the two objects before correcting the split while in Fig. 4.4 (b) we show the merged object. We merge two image objects by finding the contour point in each one that is closest to the other (i.e., points CP1 and CP2) in Fig. 4.4. Then, we draw lines from the



(a) Before correcting (b) After correcting

Figure 4.4: Example of correction of split between two objects

points before and after CP1 and CP2 connecting the two objects.

4.3.3 Validating Recovery from Occlusions

We investigated the use of invariant wavelet features to validate object matching post occlusions. To achieve this, for every VO_j, we create a new bounding box $\mathrm{BB}_{\mathrm{VO}_j}$ using

$$\mathrm{BB}_{\mathrm{VO}_j}^{width} = max(\mathrm{MBB}_{\mathrm{VO}_j}^{width}, \mathrm{MBB}_{\mathrm{VO}_j}^{height}), \tag{4.19}$$

$$\mathrm{BB}_{\mathrm{VO}_j}^{height} = max(\mathrm{MBB}_{\mathrm{VO}_j}^{width}, \mathrm{MBB}_{\mathrm{VO}_j}^{height}), \tag{4.20}$$

$$\tag{4.21}$$

where $\mathrm{MBB}_{\mathrm{VO}_j}^{width}$ is the width of the minimum bounding box of VO_j, $\mathrm{MBB}_{\mathrm{VO}_j}^{height}$ is its height, and $\mathrm{MBB}_{\mathrm{VO}_j}^{center}$ is its center. We perform this step to get a square image of the object required by the subsequent transformations. The next step is to nullify the background texture effect. To do that, let $F_n(n_1, n_2)$ denote the pixel at spatial position (n_1, n_2) and frame n of size $N_1 \times N_2$. We set to zero all pixels in $\mathrm{BB}_{\mathrm{VO}_j}$ and not in VO_j using contour filling (see Fig. 4.5), i.e.,

$$F_n(n_1, n_2) = 0, \quad \forall (n_1, n_2) \in \mathrm{BB}_{\mathrm{VO}_j} - \mathrm{VO}_j, \tag{4.22}$$

Note that we apply contour filling on the binary image of the object from segmentation, then apply the result to the normal image.

After the previous preprocessing steps, we extract the invariant wavelet features of the sub-image $F_n(n_1, n_2)$ $(n_1, n_2) \in \mathrm{BB}_{\mathrm{VO}_j}$ using [86] to obtain the feature vector for VO_j. The reason we use invariant wavelet features is to tolerate a degree of deformation in the object shape from one frame to another. The process of extracting the invariant wavelet features begins by transforming the object's square sub-image $F_n(n_1, n_2)$, $(n_1, n_2) \in \mathrm{BB}_{\mathrm{VO}_j}$ using the log-polar transform to obtain an $K_1 \times K_2$ log-polar image $lp(k_1, k_2)$. Then, we apply the discrete wavelet packet transform (DWPT) using *db*4 wavelets to $lp(k_1, k_2)$ and its one-row circular shift down version to create an oct-tree. Formally, let i denote the sub-band index (e.g., i $\in \{LL, LH, HL, LL\}$), j denote the decomposition level, ϕ denote the scaling function, and φ denote the wavelet function, the DWPT is done by

$$W_{\phi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{n_1=1}^{N_1-1} \sum_{n_2=1}^{N_2-1} lp(k_1, k_2) \phi_{\mathrm{j}, n_1, n_2}^{\mathrm{i}}(k_1, k_2), \tag{4.23}$$

$$W_{\varphi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{n_1=1}^{N_1-1} \sum_{n_2=1}^{N_2-1} lp(k_1, k_2) \varphi_{\mathrm{j}, n_1, n_2}^{\mathrm{i}}(k_1, k_2), \tag{4.24}$$

where $W_{\phi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)$ denotes the approximation wavelet coefficients at level j of sub-band i, and $W_{\varphi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)$ denotes the details wavelet coefficients at level j of sub-band i. The oct-tree is adaptively pruned based on a information cost function in order to decrease the number of computations.

$$IC(W_{\phi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)) = \sum_{n_1, n_2} ln(W_{\phi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)^2). \tag{4.25}$$

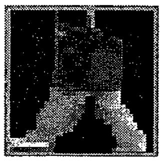


Figure 4.5: MBB expansion and nullifying background texture.

Finally, we compute energy signatures for the sub-band in the pruned tree and the most dominant (largest) signatures are used as a feature vector. We use the energy signature function

$$ES(W_{\phi}^{\mathrm{i}}(j, m_1, m_2)) = \frac{1}{N_1 N_2} \sum_{n_1, n_2} |W_{\phi}^{\mathrm{i}}(\mathrm{j}, m_1, m_2)|. \tag{4.26}$$

Post occlusion match validation works by extracting the features from occluding objects and comparing it to the extracted features post occlusion. We achieved limited success with this method as we found it to be sensitive to the object contour shape due to the fact that most of the texture information in video objects comes from the edges rather than the actual inside of the object, which tends to consist of more homogeneous than textured regions. Moreover, calculating wavelet features is far more computationally expensive than calculating color features. There exists other methods (e.g., [87]) which use directional wavelets in tracking (e.g., Gabor wavelet transform to track deformation of faces), but are still computationally expensive for multiple video object tracking compared to color features.

We investigated color as an alternative method to validate post occlusion matches. The advantage of color is that no change in the object's minimum bounding box is necessary. We investigated the same color features used by particle filter trackers [80]. These color features are calculated by finding the 16-pin histograms on the red, green, and blue channels of the object image after the background is removed using the segmentation mask. The Bhattacharyya histogram distance D_B is computed between the histograms and the maximum over the three color channels is used to calculate the probability of a match using

$$P = e^{-2D_B^2}. \tag{4.27}$$

With two object's occlusions, there are two possible matching scenarios (each consisting of two object matches) post occlusion. The probability of a matching scenario is the product of the probability of its two matches.

4.4 Results

4.4.1 Parameters and Limitations

There are four thresholds used in the proposed algorithm:

T_{age}: the minimum age (number of frames) of an object to be considered stable. For typical people monitoring applications we set $\mathrm{T}_{\text{age}} = 0.5 \times F_r$ where F_r is the frame rate in frames per second (fps).

T_{corr}: the minimum confidence value for a correspondence (match) to be considered reliable, typically $\mathrm{T}_{\text{corr}} = 0.8$. Setting T_{corr} too low will result in incorrectly accepting correspondences with split or occluded objects. Setting it too high will reject correct correspondences with objects that exhibit little change between frames.

T_{dist}: the maximum distance between two objects to consider them close enough to suspect occlusion or split. We adapt this threshold to the size of the objects and the frame rate using:

$$\mathrm{T}_{\text{dist}} = \frac{FR}{F_r} + \min\left(\mathrm{T}^{M}_{\text{dist}}, \max\left(W_i, H_i\right)\right) \tag{4.28}$$

where $FR = 250$ empirically, $\mathrm{T}^{M}_{\text{dist}} = 30$ is an upper limit for T_{dist} and W_i and H_i are the width and height of VO_i, respectively.

T_{chg}: this threshold defines the amount of change in the motion of a side that we consider a deviation from a normal behavior. This is a dynamic threshold and depends on the size of the objects:

$$\mathrm{T}_{\text{chg}} = \max\left(\mathrm{T}^{m}_{\text{chg}}, \min\left(\mathrm{T}^{M}_{\text{chg}}, 0.3 \times \min\left(H_i, W_i\right)\right)\right); \qquad (4.29)$$

where $\mathrm{T}^{m}_{\text{chg}} = 2$ and $\mathrm{T}^{M}_{\text{chg}} = 15$ are the minimum and the maximum change threshold, respectively.

The proposed method is robust in the sense that a small deviation in the mentioned threshold will not significantly affect the performance, as validated in experiments. The proposed method can handle partial, heavy and multiple occlusions, but as with all trackers, is challenged with long-lived and total occlusions and in scenes with objects of highly varying scales. Another challenging situation happens with occlusions that take place at the border of the frame as one object exits and another enters due to the disappearance of one of the objects. This goes to show the difficulty of the multiple object tracking problem and the need for further research in this area.

4.4.2 Data Set

We have conducted experiments on nine standard and four local surveillance video sequences with a total of 11125 frames to test the offline performance of the proposed algorithm. Tested videos include originally compressed video (e.g., *Comm2*, *Meet-Split-3rdGuy*, *LeftBag*). Table 4.1 summarizes the profile of the test sequences used.

Our method may give inaccurate results when objects significantly change their motion direction *during* occlusion. It is also limited when multiple objects enter the scene occluded (e.g., as a group) and they are identified as one object. When the objects eventually separate, the system will attempt to reconnect them as it interprets what happened as a split event. This is a temporary behavior since the proposed method will reinterpret the split as a real separation if the split event is

Name	Environment	Source	No. of Frames	Profile
Comm2	Outdoor	unknown	604	B
Street Survey	Outdoor	U. Rochester	1000	D
Urbi	Outdoor	COST 211	300	A, E
Road1	Outdoor	COST 211	300	A, E, F
Hall-monitor	Indoor	COST 211	300	A, F
Occlusion Vandalism	Indoor	Concordia	1300	B
Meet-Split-3rdGuy	Indoor	PETS 2004	929	A, E
Ekrlc	Indoor	U. Rochester	587	B, C, D
Ekrlb	Indoor	U. Rochester	678	B, C, D
Left bag	Indoor	PETS 2004	1444	A, E
CU Hall	Indoor	Concordia	690	B
Bishop entrance	Outdoor	Concordia	1822	A, F
INRS Hall	Indoor	INRS	655	F
CU Hall 2	Indoor	Concordia	514	A, F

A – Occlusions	C – Heavy occlusions	E – Small objects
B – Complete occlusions	D – Multiple occlusions	F – Split

Table 4.1: Profile of test sequences used.

consistent in time and the distance between the objects is increasing.

4.4.3 Objective evaluation

To measure the performance of the tracking algorithm objectively, we first use the "motion difference along object boundaries (MDOB)" defined in [88], Section II.C. The MDOB quantifies how well the estimated object boundaries coincide with actual motion boundaries:

$$\text{MDOB} = \frac{\sum_{i=1}^{Q} \delta_i \cdot w_i}{\sum_{i=1}^{Q} w_i}, \quad \delta_i = \exp\left(-\frac{\|\mathbf{v}_i^O - \mathbf{v}_i^I\|}{\sigma^2}\right), \tag{4.30}$$

where Q is the number of points in the boundary, δ_i measures the difference between motion vectors inside ($\mathbf{v}_i^I$) and outside ($\mathbf{v}_i^O$) the object at point i of the boundary, and w_i measures the reliability of the estimated motion vectors. Fig. 4.6 shows the MDOB measures for the first 114 frames of the *Street Survey* sequence for the proposed algorithm and the region-based algorithm [8] as a reference. The average of the MDOB measure for the first 100 frames of this sequence during which three objects are heavily occluded is around 0.5 for the proposed method and around 0.9 for the reference algorithm [8]. In this plot a smaller value indicates that the estimated object boundary coincides with a real motion boundary, meaning that a smaller value represents a better performance. It can be seen that the proposed algorithm outperforms the referenced method.

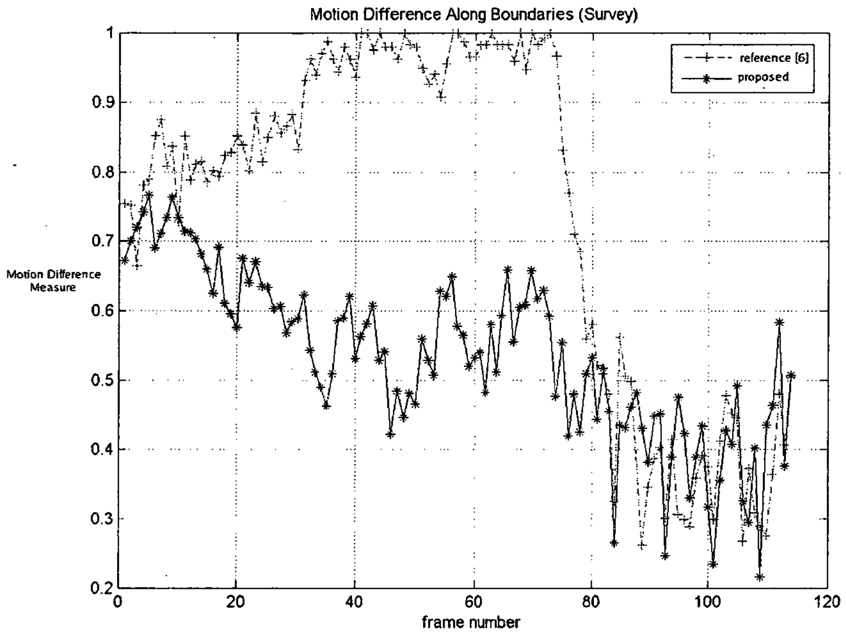


Figure 4.6: Motion difference along object boundaries (MDOB) measure during the first 114 frames of the *Street Survey* sequence for proposed and reference methods [8]. A lower value indicates better performance.

We also used two other objective measures: the Average Size Detection Rate (ASDR) and the Label Tracking Detection Rate (LTDR) defined in [89], to compare the proposed algorithms with the algorithm in [10]. This algorithm identifies and handles long-lived occlusions by creating a group that contains the information from the occluded objects. Once the occlusion is finished the algorithm identifies a split and correctly label the separated objects. This approach is less valid for long-term partial occlusions such as when two objects enter the scene. occlude, and then continue to walk or interact with each other for an extended period of time. An example of this scenario is given in Fig. 4.7 showing the tracking results for the *Meet_Split_3rdGuy* sequence in which two people meet on a hall with very challenging illumination conditions and small objects. Fig. 4.7(a) shows the objects before the occlusion starts. Fig. 4.7(b, c, d and e) show how the detection and correction of the occlusion by the proposed algorithm leads to continuity of tracking during occlusion. Fig. 4.7(f) shows the tracked objects after recovery from occlusion. We cannot guarantee the successful tracking of the group of objects after they occlude if the occlusion takes a lot of time, especially with other objects getting involved. The continuous detection and correction of occlusions and splits allows us to continue to have information about the last known position of an object in case it is lost, which increases the chances of its successful recovery.

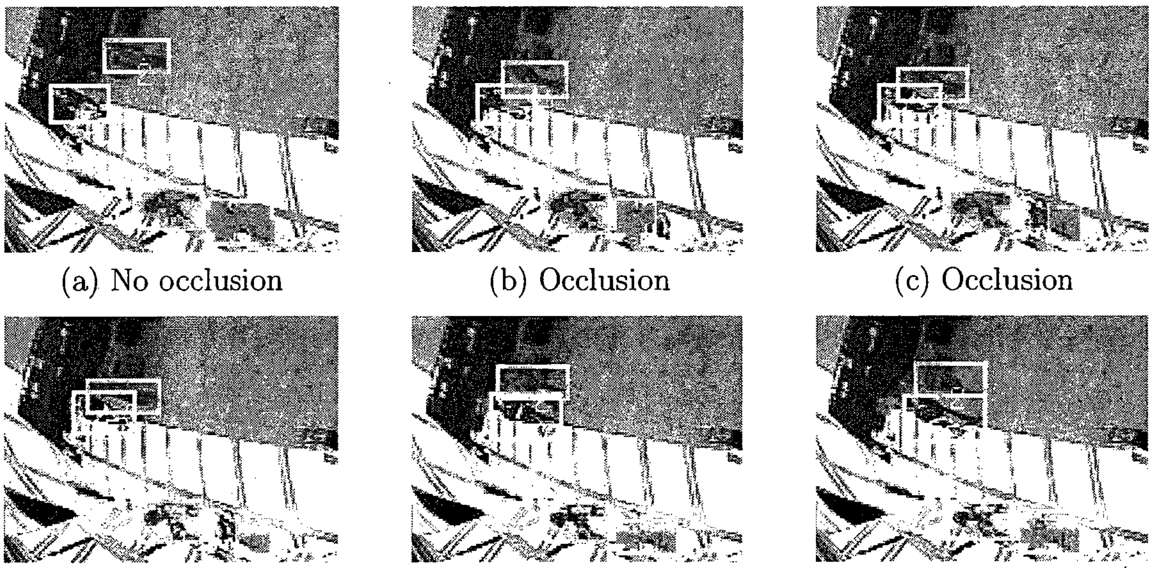
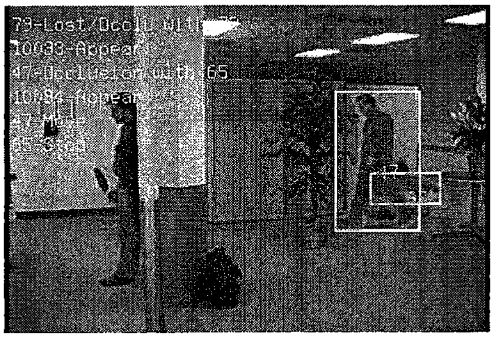
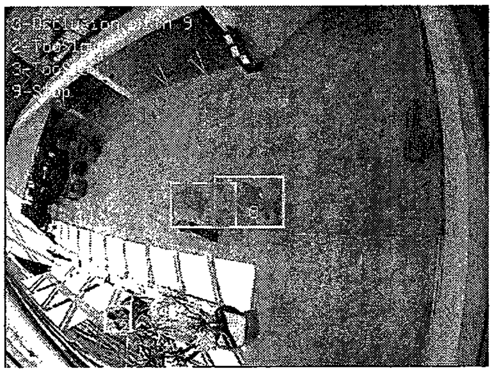

(a) No occlusion (b) Occlusion (c) Occlusion
(d) Occlusion (e) Occlusion (f) Recovery

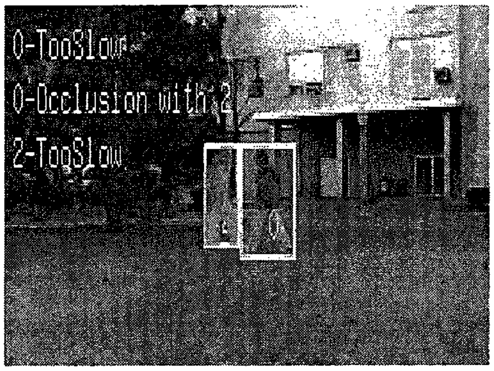
Figure 4.7: Occlusion detection and correction results. *Meet_Split_3rdGuy* sequence.

We used the sequences *Ekrlc*, *Left bag*, *Comm2*, and *Street Survey*, for which the ground truth segmentation and tracking information is available and a representative frame is shown in Fig 4.8. Note that ASDR and LTDR are in [0, 1] with 0 indicating bad performance and 1 indicating a perfect performance.

Objective results are given in Table 4.2. Both algorithms perform very well at


(a) Ekrlc


(b) Left bag



(c) Comm2

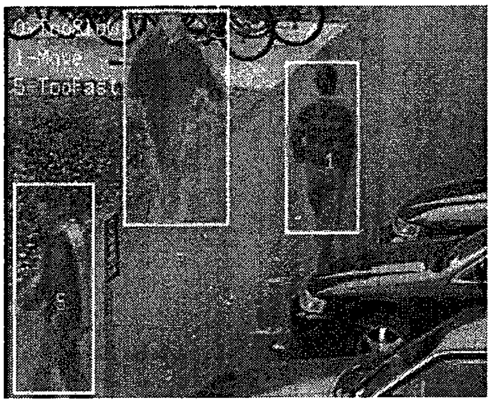

(d) Street survey

Figure 4.8: Representative result frames from test sequences in Table 4.2.

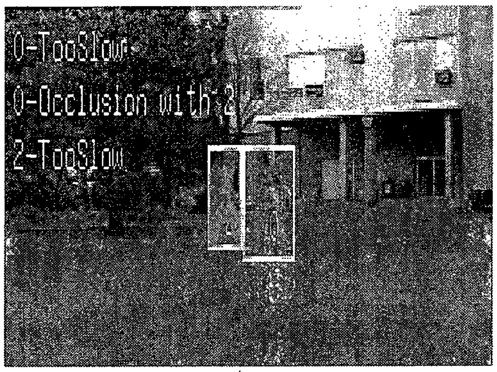
tracking the size of the objects (ASDR). The proposed algorithm outperforms the referenced in the label tracking detection rate (LTDR) measure, which indicate that the occlusions and split where solved in a successful way.

	Proposed		Reference [10]	
Video	ASDR	LTDR	ASDR	LTDR
Ekrlc	0.967	0.658	0.875	0.407
Left bag	0.993	0.983	1.000	0.968
Comm2	0.997	0.973	0.995	0.948
Street Survey	0.949	0.845	0.915	0.796
Average	0.9765	0.8647	0.9462	0.7797

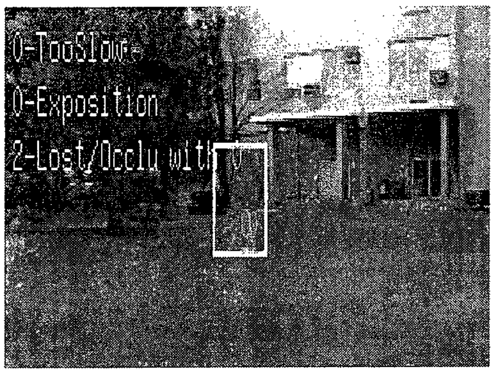
Table 4.2: Comparison of the proposed and the referenced [10] algorithms.

4.4.4 Visual results of occlusion detection and correction

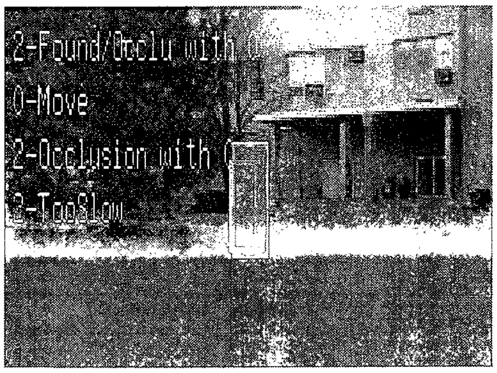
In Fig. 4.9, the occlusion detection and correction is shown for the *Comm2* sequence in which two objects completely occlude. Fig. 4.9(a) shows the tracked objects before the occlusion takes place. The occlusion starts in Fig. 4.9(b). The proposed algorithm detects and corrects occlusion and enables the tracking of objects during heavy occlusion as shown in Fig. 4.9(c). Fig. 4.9(d) shows the objects after recovery from occlusion.



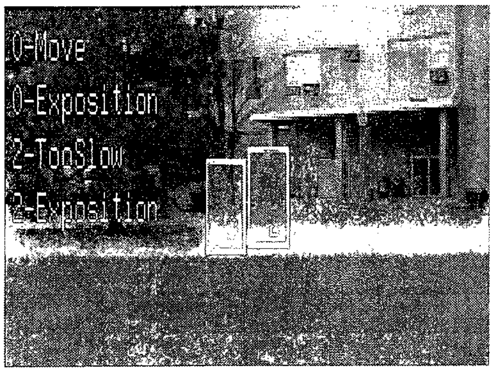
(a) No occlusion



(b) Heavy occlusion



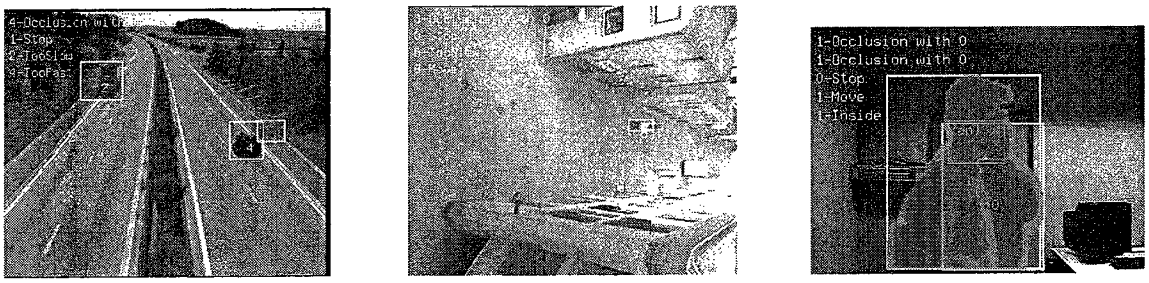
(c) Heavy occlusion



(d) Recovery

Figure 4.9: Occlusion detection and correction results. *Comm2* sequence.

Fig. 4.10 shows one significant frame from each of the other sequences used to test the algorithm. We have chosen a frame were the occlusion event is clearly identified.

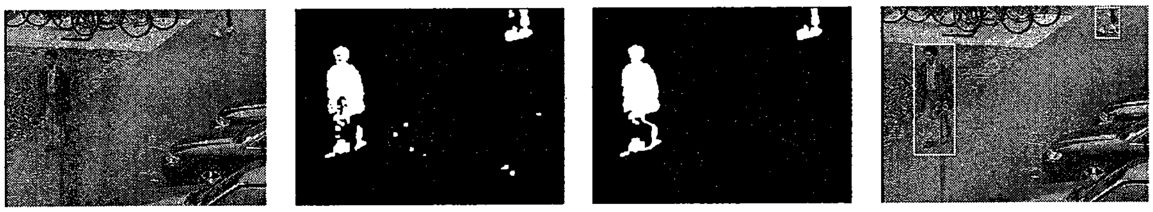


(a) Road1 (b) Urbi (c) Occlusion Vandalism

Figure 4.10: Representative result frames from remaining test sequences of Table 4.1.

4.4.5 Visual results of split detection and correction

An example of multiple split correction is shown in *Street Survey.* The change detection returns three fragments of one object due to color similarity as in Fig. 4.11(b). Split is detected and corrected in Fig. 4.11(c). Fig. 4.11(d) shows the improvement achieved due to split correction.



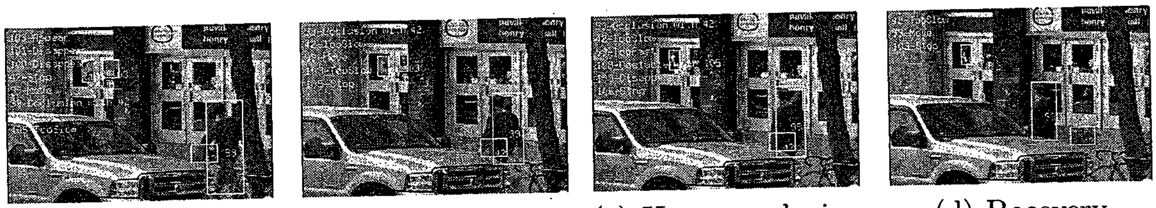
(a) Original (b) Change detection (c) Split correction (d) Object tracking

Figure 4.11: Split detection and correction results. *Street Survey* sequence.

4.4.6 Visual results using online captured and processed video sequences

Fig. 4.12 and 4.13 show sample results using the surveillance system in outdoor and indoor environments. As can be seen, the proposed algorithms correctly detected and solved the occlusion events present on these sequences. Note that in these examples the online output sequences are shown in color and labels identifying major events, the main output from the online surveillance system, are also shown.

In the first example shown, we applied our algorithm to a sequence captured at one entrance to a building of Concordia University. In this sequence a bag deposited near the entrance in Fig. 4.12(a) is first occluded and then completely occluded by a passing pedestrian in Fig. 4.12(b) and Fig. 4.12(c) respectively. The system never loses the identity of the bag and the shapes of the two objects are fully recovered in Fig. 4.12(d).

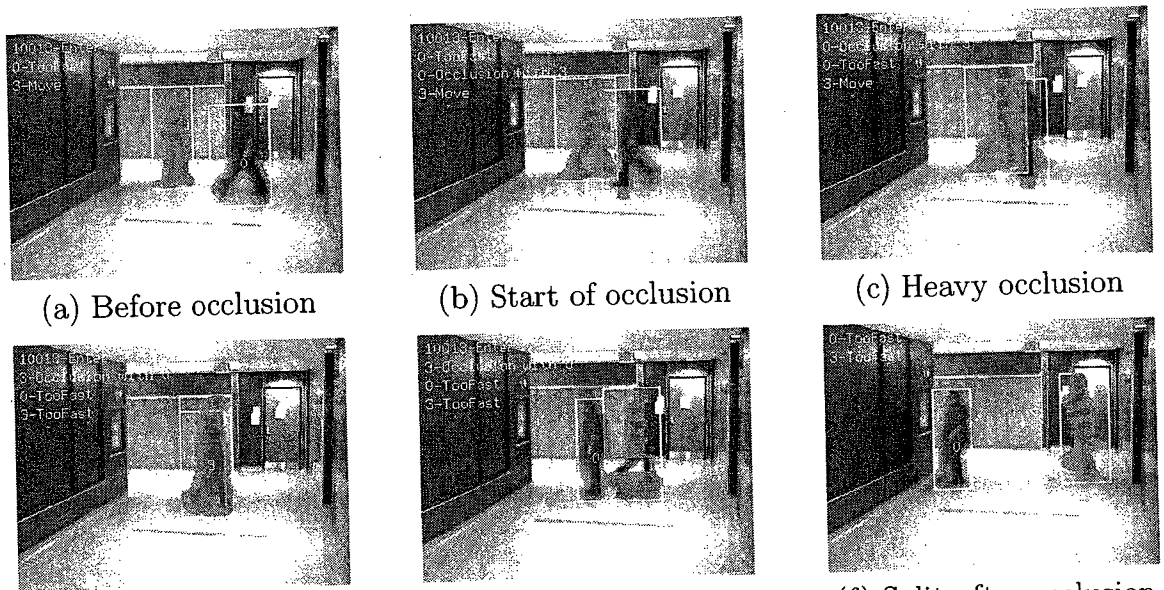


(a) Start of occlusion (b) Heavy occlusion (c) Heavy occlusion (d) Recovery

Figure 4.12: Occlusion detection and correction. Online *Bishop Entrance* sequence.

The second example shows how the online testing system correctly tracks two people crossing paths in a hall. The two people are completely separated in Fig. 4.13(a). They start to occlude each other in Fig. 4.13(b) and the occlusion advances until it becomes a heavy occlusion in Fig. 4.13(d). The system correctly tracks each object once the heavy occlusion is finished in Fig. 4.13(d) and the occlusion is finalized in Fig. 4.13(e) where the two people are again completely separated. Note that the system simultaneous handles occlusion and split, e.g., in Fig. 4.13(d) due to the similarities between the man walking to the right and the background the result from the segmentation algorithm consist of two objects but incorrectly segmented; the system is still able to detect and correct occlusion and split.

We show in Fig. 4.14 the result of testing the color features for post occlusion match validation. We show eleven occlusion scenarios in the *CU Hall* video sequence. Each row shows the matching scenarios along with its probability. We can see from Fig. 4.14 that in all matching scenarios, the probability of the correct match is higher



(a) Before occlusion (b) Start of occlusion (c) Heavy occlusion

(d) Complete occlusion (e) Split during occlusion (f) Split after occlusion

Figure 4.13: Occlusion detection and correction results. Online *CU Hall* sequence.

than the wrong match. This leads us to conclude that color features are more robust to changes in object shape and pose during occlusion than texture features and are far less computationally expensive. Nevertheless, more research is needed to increase the difference between the probabilities of true and false matching scenarios.

4.5 Summary

This chapter presented an algorithm for the real-time detection and correction of occlusion and split in object tracking by monitoring sudden changes in the spatio-temporal features of objects. The proposed algorithm proves useful for surveillance applications because of its: 1) good performance in multiple and heavy (or total) occlusion; 2) low computational complexity; 3) distinction between real split and split due to faulty segmentation; and 4) handling of simultaneous occlusion and split. These characteristics were demonstrated using standard surveillance video sequences containing multiple and heavy occlusions, as well as splits due to faulty segmenta-

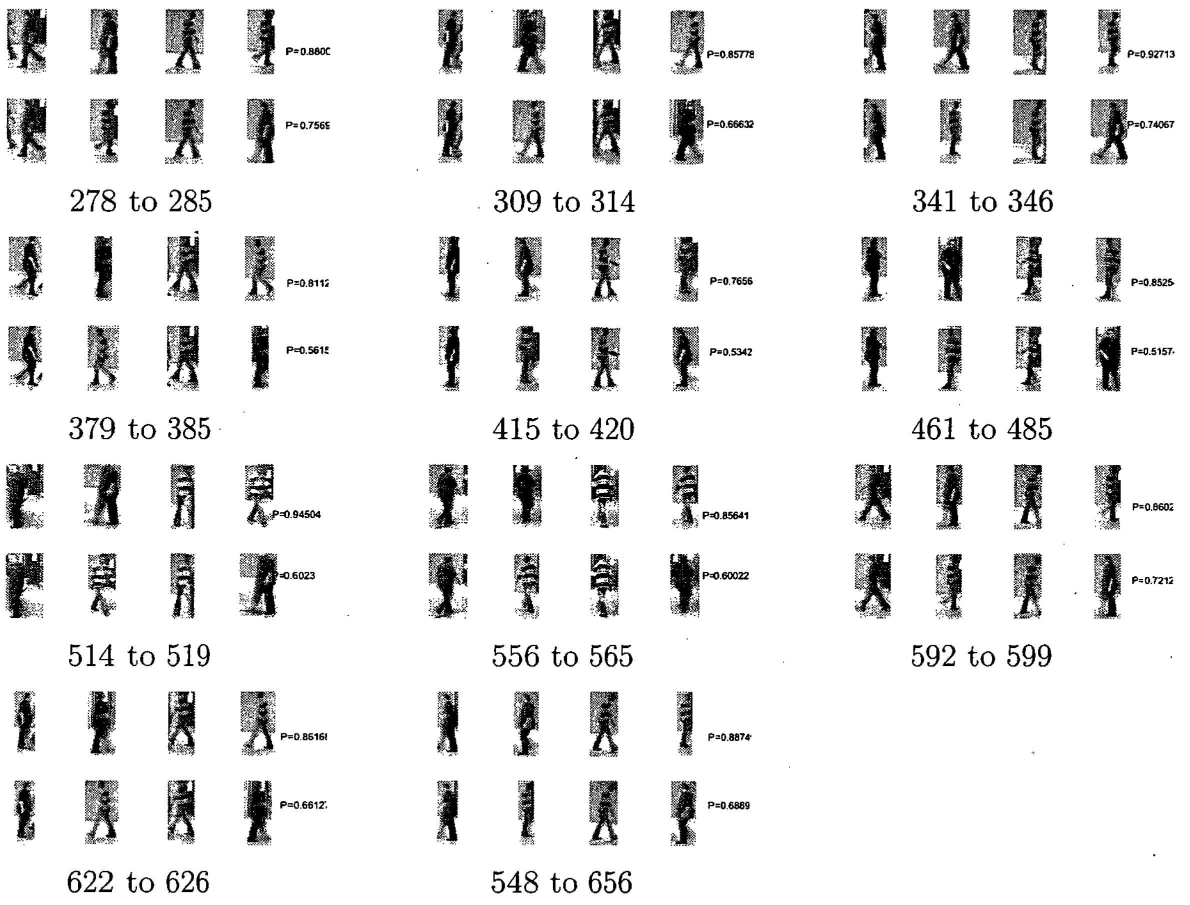


Figure 4.14: Post occlusion match scenarios for 11 cases of occlusions in the *CU hall* sequence and their probabilities. The probabilities of the correct matches using color features are higher than the wrong matches in all incidents.

tion. The proposed occlusion and split handing methods improve the robustness to segmentation errors in tracking. They still, however, require tracking to be temporally stable as discussed in Section 3.3, which requires segmentation to be temporally stable. Temporally unstable tracking leads to difficulty in creating matches between image and video objects due to major deviations in object features. Robust image and video object matching is a corner stone to successful tracking.

Chapter 5

Vandalism Detection

5.1 Introduction

This chapter proposes a novel method for the detection of vandalism events in video sequences. The method is based on a proposed definition for common vandal behaviors recorded on surveillance video sequences. To do this, the method monitors changes inside a restricted site containing vandalism-prone objects such as a vending machine, a pay phone, or a street sign. When an object is detected as leaving such a site, the proposed method checks if the site contains temporally consistent and significant static changes, representing damage. If there are such changes and given that the site is normally unchanged after legal use, a vandalism event is declared and the vandals are tracked. The proposed method is tested on video sequences showing real and simulated vandal behaviors and it achieves a detection rate of 96%. It detects different forms of vandalism such as graffiti and theft, and can handle sudden illumination changes, occlusions, and segmentation errors.

5.2 Review of Related Work

Vandal behaviors are defined in [12] as unlawful destructing or damaging of public or private property. According to the *Canadian Centre of Justice Statistics*, vandalism in 2004 accounted for up to 36% of all reported crimes [11]. In the year 1991, over two million incidents of vandalism against private property have occurred in the UK [12]. The cost of vandalism is not only financial, but also social. According to a poll for *The Times*, vandalism is regarded by people as one of the most important problems facing their families [12].

The growth of visual information has created an urgent need for effective methods for analyzing visual information in the security domain [90]. The deployment of "intelligent" video surveillance systems [91] able to detect and report vandalism as it happens is, therefore, becoming popular. There is a growing trend of fitting public transport vehicles (e.g., buses [92], trains [93] and taxis) with surveillance cameras for the purpose of human behavior recognition [94], and eventually having a city-wide surveillance network for detecting vandalism [13]. Deploying video-based automatic vandalism detection systems in monitoring and preserving archaeological sites [95] is also popular.

The automatic detection of vandalism in video surveillance, is a challenging task because of: 1) the complex and unpredictable nature of a vandalism act and the speed at which it may occur; 2) the underlying difficulty of finding a unique definition for vandalism which may vary based on social contexts and applications; 3) the difficulty in distinguishing between normal and vandal interaction between persons and vandalism-prone objects or sites; and 4) the lack of real vandalism test video sequences publicly available for testing.

Few methods have been presented in the literature for the detection of vandalism. They can be categorized into single camera [96–100] and multiple camera methods [101, 102]. The method in [97] defines a semi-automatic system to aid an operator in the detection of vandal acts. The actual detection is done by feeding geometric and kinematic features to a neural network classifier which is trained to recognize suspicious variations in those features. Some of these features include the coordinates of the minimum bounding box and center of gravity of tracked objects along with their areas and perimeters. This approach requires training the classifiers and expects vandals to follow certain behaviors. The method in [98] detects vandalism by distinguishing between moving objects and expanding static objects which are caused by vandalism. The distinction is made using a mean change detection image. The challenge is in objects moving in the camera direction which may be confused with graffiti. The method in [99] detects movement and faces of passengers in public transport vehicles for monitoring of vandal behaviors. It assumes that vandals in such situations move abnormally (e.g., actively switch seats or have large body motion gestures) relative to other passengers. This approach is specific to public transport vehicles in which passengers are normally not moving (e.g., standing or sitting). The method in [100] uses a time of flight camera which generates distance data to detect graffiti.

Methods using multiple cameras to detect vandalism avoid dealing with occlusion and can detect vandalism in larger areas. The authors of [101] use multiple cameras and trackers to build a combined graph of the positions and dimensions of persons in the scene. The combined graph is then used to recognize vandalism using a temporal constraint network. The method in [102] distinguishes between temporal and spatial changes using multi-view change detection.

In this chapter, we propose to detect vandalism by monitoring and evaluating changes inside predefined restricted sites as objects enter or exit these sites. The advantages of the proposed method are: 1) detecting vandalism without training; 2) detecting different forms of vandalism such as damage, graffiti and theft (or removal); 3) detecting and tracking vandals; 4) handling changing illumination conditions, occlusions, and segmentation errors; and 5) fast approach with high detection rate.

The remainder of the chapter is organized as follows. Section 5.3 presents the proposed vandalism detection methods theoretically. Simulation results are discussed in Section 5.4, and Section 5.5 summarizes the chapter.

5.3 Proposed Algorithm for Vandalism Detection

5.3.1 Defining Vandalism

We identify vandalism by detecting permanent changes inside a particular region of the frame that has been previously classified as background and is of interest from the point of view of surveillance. Formally, we define *vandalism* as a *video event* instantiated by a *video object* which inflects temporally consistent static changes (e.g., damage) inside a predefined restricted region that is supposedly unchanged by normal (i.e., legal) interaction with video objects. This temporally consistent change inside the restricted site in the definition is identified, for example, as a newly appearing video object satisfying certain conditions such as being static for a long time while inside the site.

A *video object* refers to a temporally consistent region (over a short period) in a video sequence. Video objects have spatio-temporal features such as contour, area, motion, and trajectory. We detect and track video objects and use information about their features. For example, a video object has a unique identifier (ID) maintained by the tracking algorithm during the life-time of an object in the video sequence. A *video event* is an interpreted spatio-temporal relationship associating one or multiple objects (e.g., *moving*, *stay long*, and *is inside*). Video events have information associated with them such as the IDs of the video objects involved in the event, the time at which it is detected, and its duration which is the number of consecutive frames the event is detected.

We only consider rigid vandalism-prone objects that do not change over time. This includes pay-phones, vending machines, and paying stations in parking lots. For example, vandalism of electronic street signs switching content periodically is not considered. Also, we expect that the vandalism act alters the normal appearance of objects. Meaning, after the site is vandalized, there is visible damage (i.e., change) to the site. We use video object segmentation and tracking. There is no restrictions relating to which algorithms are used as long as they provide the information necessary for the proposed algorithm.

5.3.2 Defining Vandalism Targets

We define restricted (i.e., vandalism-prone) sites by marking them in the scene as in Fig. 5.1 (the white box encapsulating the street sign). This approach allows us to restrict the vandalism detection process to the important elements in the field of view of the camera and to reduce alerts for changes to non-important parts of the image. This is what we call surveillance targets. There is no restriction on the number of vandalism-prone sites in the same scene. The proposed algorithm is designed to handle the detection of simultaneous vandalism events. This is helpful, for example, in detecting which street phone or vending machine is vandalized in a scene where a number of restricted sites are monitored by the same camera. On the other hand, defining vandalism targets avoids unnecessary vandalism checks and significantly reduces false alarms. The proposed method not only detects vandalism, but also vandals. The detection and tracking of vandals is a key advantage. It can be used to track the vandal to identify, for example, where the vandal has existed the room or ally. Note that when changing the context associated with the restricted site from a vandalism-prone site to an object abandoning site, the proposed method can be used to detected abandoned objects.

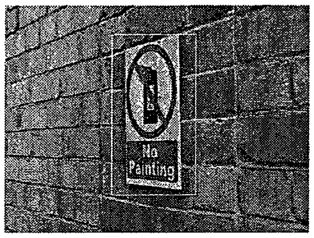


Figure 5.1: A predefined restricted region R_r with a rectangular box.

We discriminate between all detected changes inside the restricted site and decide if said changes are from normal usage of the site or due to vandalism. These changes result from change detection between the current frame and the background frame because we consider the restricted site as part of the background and is left essentially in the same state before and after normal use of the site.

The proposed algorithm for vandalism detection is triggered by the event of a video object entering (or being partially inside) a restricted site and is confirmed when the video object exits (or is completely outside) a restricted site. We will discuss each one of the these two events individually in Sections 5.3.3 and 5.3.4, respectively.

5.3.3 Detecting vandalism triggered by enter event

Let R_r denote a predefined restricted site with the identifier r, and VO_i denote a video object with the identifier i which has just been detected inside R_r. Formally, R_r and VO_i can be viewed as sets of all spatial locations of pixels belonging to R_r and VO_i, respectively. The detection of video event I_{r_i} signaling that VO_i is inside or entering R_r is done using

$$I_{r_i} : R_r \bigcap VO_i \neq \phi, \tag{5.1}$$

where ϕ is the empty set. This indicates that we initiate vandalism detection even if a vandal does not fully enter R_r (e.g., a person extending hand into R_r as in Fig. 5.8(a)). The removal of small or temporally unstable objects by the segmentation algorithm prevents (5.1) from triggering a vandalism check for erroneous objects due to noise. In (5.1), we check if there is a number of spatial locations which are common to both VO_i and R_r. Alternatively, I_{r_i} can be established by checking the area of the overlap rectangle between the minimum bounding boxes of VO_i and R_r.

Detecting I_{r_i} is the first stage in the proposed method. It may later evolve to a vandalism event if we detect damage in R_r. I_{r_i} is verified with every new frame in order to determine if it is a normal interaction between VO_i and R_r or a vandalism event (i.e., N_{r_i} or V_{r_i}). We monitor VO_i while inside R_r to detect one of two possible scenarios. The first scenario is that I_{r_i} turns out to be N_{r_i} signaling normal use of the restricted region R_r when VO_i is moving, i.e.,

$$\mathrm{N}_{r_i} : \mathrm{I}_{r_i} \wedge \mathrm{M}_i^l, \tag{5.2}$$

where M_i^l denotes the event that VO_i is moving at time instant l. In this case, the information related to I_{r_i} is updated with the current position of the object and the new frame number l, and the process continues by advancing to the next frame. Event M_i^l is established by examining the average of the motion of VO_i in the past K frames. Meaning, M_i^l is declared as

$$\mathrm{M}_i^l : \frac{1}{K}\sum_{k=1}^{K} \|\mathrm{M}_i^{l-k}\| > \mathrm{t}_M, \tag{5.3}$$

where $\|\mathrm{M}_i^{l-k}\|$ denotes the magnitude of the motion vector of VO_i in frame $l-k$ (l being the current frame number), and t_M is a threshold for how much apparent motion constitutes real motion. We adapt K to the frame rate using $K = \frac{F}{5}$, where F is the frame rate. t_M is adapted to the video frame size using $\mathrm{t}_M = W/160$, where W is the frame width. We characterize the motion of the video object by the displacement vector of the center of the minimum bounding box. It is also possible to use other methods for representing or estimating the motion of objects [103] or alternative definitions of M_i^l (e.g., with more robust estimators than the mean).

In the second scenario, VO_i stops moving while I_{r_i} is still declared and this situation is persistent over a long time. The proposed method detects this as an alarming case of possible vandalism or *staying long in a restricted site* L_{r_i} while being static. Although this event is not a clear case of vandalism, we declare a vandalism event because there is a permanent change to the restricted site, and this is an alarming situation. This could happen for example of an object without entering the scene throws another object to inflect damage on the vandalism-prone site (e.g., breaking glass with a stone). Event retrieval can be used to decide later on the course of action based on the nature of the damage (permanent change) detected. This is governed by

$$\mathrm{L}_{r_i}: \ \mathrm{I}_{r_i} \wedge \mathrm{S}_i \wedge \left(\frac{|\mathrm{VO}_i \bigcap \mathrm{R}_r|}{|\mathrm{R}_r|} > \mathrm{t}_c \right), \tag{5.4}$$

where $\frac{|\mathrm{VO}_i \bigcap \mathrm{R}_r|}{|\mathrm{R}_r|} > \mathrm{t}_c$ states that the area of the overlap between VO_i and R_r is greater than t_c% of the area of R_r, t_c is the percentage of change in R_r which constitutes an alarming level of changes. We set t_c to 0.5, which is the default value. t_c can also be set to an exact value for theft applications, because the size of the object monitored for theft (or removal) is known at the time of defining R_r. S_i is the event that the video object VO_i has stayed at the same location for a considerably long time. To detect S_i, we use

$$\mathrm{S}_i: \ \overline{\mathrm{M}}_i^l \wedge \left(\left| \overline{\mathrm{M}}_i^l \right| > \mathrm{t}_T \right) \wedge A, \tag{5.5}$$

where $\overline{\mathrm{M}}_i^l$ is the complement or negation of event M_i^l indicating that VO_i is static, $\left| \overline{\mathrm{M}}_i^l \right|$ represents the duration of event $\overline{\mathrm{M}}_i^l$ (i.e., the number of consecutive frame the object has been static), and t_T is a threshold that defines the minimum number of frames that an object must be in the same location to be considered static for a long time, which we adapt to the frame rate FR using $t_T = 4 \times FR$. A is the event that object area is stable. We add it to avoid confusing an object that is not moving in the camera direction as static. We use the mode on the sign of the area derivative to establish the A event. We check when the mode is zero, meaning the area is not increasing or decreasing. First, the discrete temporal difference of area $\Delta\mathrm{A}_i^l$ is calculated using

$$\Delta\mathrm{A}_i^l = \mathrm{A}_i^l - \mathrm{A}_i^{l-1}. \tag{5.6}$$

where A_i^l is the area of the video object VO_i at the current frame l. We use K consecutive differences and create the set $Q = \{\mathrm{sgn}(\Delta\mathrm{A}_i^k)\}$, where $k \in \{l, l-1, ..., l-K\}$, and sgn is the sign function. The event A is defined using

$$A: \quad mode(Q) = 0. \tag{5.7}$$

After triggering vandalism detection, the process advances to the next stage of confirming vandalism. Note that it is not difficult to discriminate between stationary and moving objects or detecting objects that stay for long time at the same location since while detecting vandalism we are mostly dealing with humans that move relatively slowly with respect to standard frame rates (e.g. 30 frames per second).

5.3.4 Confirming vandalism detection by exit event

Once the video object VO_i has left R_r there are three possible events to consider. The first event is that there is no change to R_r, i.e., there exists no object VO_c inside R_r (indicating that R_r has not changed). This leads to confirming normal behavior E_r and $\bar{\mathrm{I}}_{r_i}$ and is governed by

$$\mathrm{E}_r : \left(\mathrm{O}_j \bigcap \mathrm{R}_r = \phi\right) \quad \forall j. \tag{5.8}$$

In this case, the system is returned to its initial state.

The second possible event happens when an O_j enters or is inside R_r as object VO_i leaves it as in

$$\mathrm{N}_{r_j} : \ \bar{\mathrm{I}}_{r_i} \wedge \mathrm{I}_{r_j} \wedge \mathrm{M}_j^l. \tag{5.9}$$

If the newly detected object O_j is moving (i.e., M_j^l), we release the video object VO_i from any responsibility. Vandalism detection as in Section 5.3.3 is then restarted for O_j.

The third event is confirming vandalism and detecting the vandal (i.e., a complete vandalism event) when there is at least one object VO_c inside R_r which is static and its area $|\mathrm{VO}_c|$ is significant with respect to R_r. This case is governed by the conditions

$$\mathrm{V}_{r_{ic}} : \quad \bar{\mathrm{I}}_{r_i} \wedge \mathrm{I}_{r_c} \ \wedge \overline{\mathrm{M}}_c^l \wedge \left(|\mathrm{VO}_c| > \mathrm{t}_c \times |\mathrm{R}_r|\right). \tag{5.10}$$

where $\overline{\mathrm{M}}_c^l$ is the event that VO_c is not moving and $\mathrm{V}_{r_{ic}}$ denotes the vandalism event with r identifying the restricted site, i the ID of the vandal object, and c the ID of the damage object. Equation (5.10) tests if a significant static change VO_c is detected inside the site after VO_i has left. This indicates that there is damage to the region R_r and since VO_i was the last object visiting the site, it is held responsible for this damage. Fig. 5.2 shows the state diagram of the proposed algorithm. Vandalism detection is started when an object is detected inside the restricted area. If the object leaves with no other objects inside the restricted area, normal operation is assumed. If the object stays for too long or leaves behind stationary objects, vandalism is declared. If the object leaves while another enters the site, vandalism detection is restarted for the entering object.

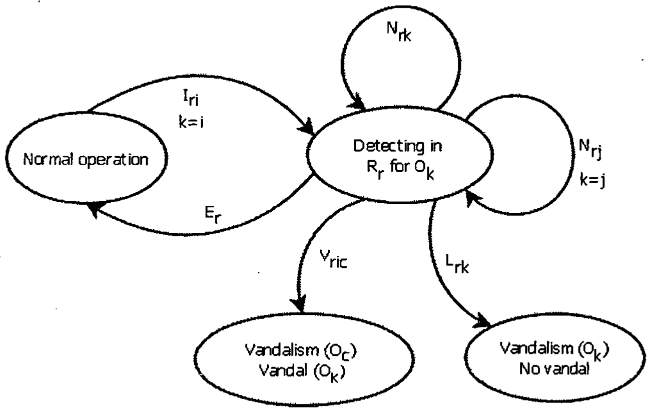

N_{rk}
I_{ri}
$k=i$
Normal operation
Detecting in R_r for O_k
N_{rj}
$k=j$
E_r
V_{ric}
L_{rk}
Vandalism (O_c) Vandal (O_k)
Vandalism (O_k) No vandal

Figure 5.2: State diagram of the proposed vandalism detection algorithm.

5.3.5 Multiple Vandals

So far, we develop the conditions for vandalism with a single object VO_c representing the change in R_r. It may happen that the change in the site is broken into multiple objects as in Fig. 5.3(d). In this case, the vandalism event includes the IDs of all these objects as shown in Fig. 5.3(d). It may also be the case that multiple objects (or persons) are vandals. If an object enters R_r with multiple objects close by, and vandalism is detected after the objects leave, then *all* objects are labeled as candidate vandals or potential witnesses as in Fig. 5.9. With the proposed multiple object tracking technique, candidate vandals are tracked until they exit the scene, which provides information on their trajectory to identify where they exited the scene. If an object is passing by during an event of vandalism, when vandalism is not confirmed yet, it is not labeled as a candidate vandal as in Fig. 5.7.

5.4 Results

5.4.1 Parameters and Limitations

The proposed method uses three thresholds, a frame-size adaptive t_M as in (5.3), a frame-rate adaptive t_T as in (5.5), and t_c as in (5.4). We give the used value or adaptation where they are defined. t_c is the percentage of change detected in the restricted site which constitute an alarming level of change. The default value is 0.5 or 50%. t_c can also be set to an exact value for theft applications, because the size of the object monitored for theft is known at the time of defining R_r. In the sequence in Fig. 5.4, the quantization threshold for the segmentation algorithm in [68] are lowered to 80, 100, and 120 due to compensate for the low illumination. The g_{lim} is changed to the low setting of 31. The upper limit of the distance threshold for split detection is lowered for the sequence in Fig. 5.3 to 10 to account for zoomed view.

Our simulations show that the proposed method is effective in detecting different forms of vandalism, e.g., caused by damage to the site (e.g., graffiti) or by removing an object from the site (e.g., theft). In both cases, the proposed method detects vandalism. In other words, the proposed method and definition of vandalism cover theft as well as long as it leads to temporally consistent change in the restricted site. The definition of vandalism based on detection of permanent change to a particular section of the frame is less dependent on the segmentation and tracking algorithms than vandal detection and tracking. As far as there is a visible change to an static object it is detected as vandalism. The robustness of segmentation and tracking adds robustness to vandal detection and tracking. We demonstrate this in Section 5.4.4. While the proposed algorithm can handle multiple objects, crowded scenes (e.g., in the street during the day) may lead to false alarms. Usually, vandalism, being a crime, occurs in a non crowded scene. The chances of false alarms in crowded scenes decreases if the expected level of damage (e.g., size of restricted area) is disproportional to the sizes of object and increases if they are similar. A possible approach to significantly reduce false alarms in crowded scenes where the size of damage cannot be anticipated is to use damage object recognition or classification. When investigating vandalism, the damage object post-vandalism is classified and if its class is found to belong to common objects like humans or vehicles, vandalism is not declared.

5.4.2 Offline Testing

The proposed method was tested on both offline videos showing real and simulated vandal behaviors, and using an online surveillance system with a static camera (Section 5.4.3). Few real vandalism test video sequences are publicly available. Combining both online and offline test sequences, we used a total of 50 different incidents of real or simulated vandalism to obtain a statistically significant detection rate. Note that sequences in Fig. 5.3 and Fig. 5.4 are originally compressed.

Figs. 5.3-5.7 show the five video sequences used in offline testing scenarios to evaluate the performance of the proposed method. In each of the figures, we show annotated video frames before, during and after the event of vandalism. The annotation carries information about the detected events and the tracking IDs of involved objects. In all the figures, frame 1 shows the bounding box of the restricted area.

Fig. 5.3 shows an outdoor video sequence publicly available in which a hand spray paints over a street sign to vandalize it. The hand movement is fast and the amount of change caused by vandalism is high relative to the size of the vandalizing object. The annotation in frame 108 reads *"0-Vandalism With 2 5"* indicating object 0 (hand) vandalizing the restricted area and the result of vandalism are objects 2 and 5.

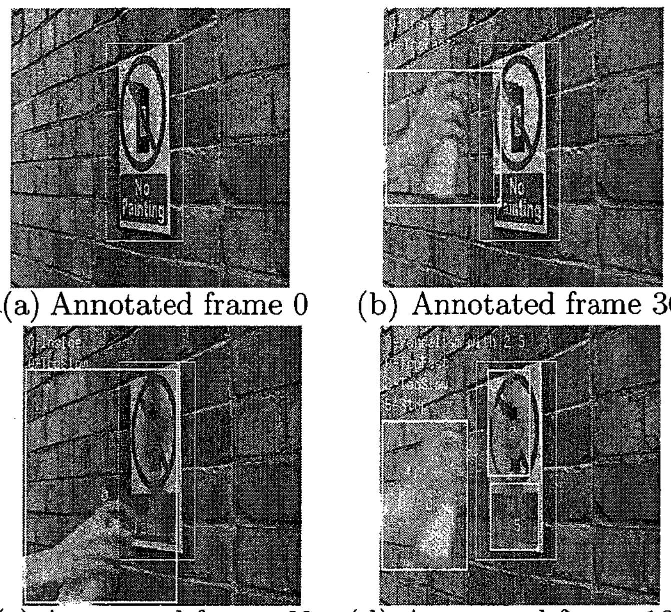
Fig. 5.4 shows another outdoor video sequence publicly available in which a vandal removes a large wall sign in bad illumination conditions and throws it on the floor. For this sequence, we also show the unprocessed video for clarity. Frame 3602 shows the vandal abandoning the site after vandalism. The annotations in in frame 3291 read *"110-Vandalism with 68"* and *"111-Vandalism with 68"* suggesting that objects 110 and 111 vandalized the restricted area resulting in object 68.

Fig. 5.5 shows two different video sequences (one in each column) captured in our lab. In Fig. 5.5(a) an actor was asked to approach and interact normally with a model pay phone hanged on a wall, whereas in Fig 5.5(b) another actor was asked to tamper with the model pay phone. The annotation in Fig. 5.5(b) frame 643 reads *"15-Vandalism with 0"* whereas there is no annotation in Fig. 5.5(a). This indicates the successful distinction between normal interaction and vandalism.

Fig. 5.6 shows the outcome of a vandalism detection check after a sudden change in illumination caused by turning on the lights. The background maintenance updated the background model and the new model is used to continue to retain the performance of the fast segmentation method and correctly track objects and detect vandalism.

Fig. 5.7 shows vandalism detection in a scene with severe occlusion. An event of complete occlusion occurs during the vandalism act. The proposed method continues to track the object correctly until it is heavily occluded. This is when object 0 is briefly lost. The object is recovered shortly and tracking continues during the remainder of the occlusion event. Vandalism is detected as the vandal leaves the restricted site.

As can be seen from Figs. 5.3-5.7, the proposed method is able to detect vandalism in all scenarios and can successfully distinguish between vandal and normal acts. The proposed method requires on average 74 ms to process a 320×240 frame (including background update, segmentation, and tracking) when implemented using C++ and run under an Intel i7 CPU 2.67GHz machine operated by Linux. Thus, the effective frame rate is about 13 frames/second. This was measured on the all offline video sequences in Fig. 5.3-5.7 totaling 6850 frames.



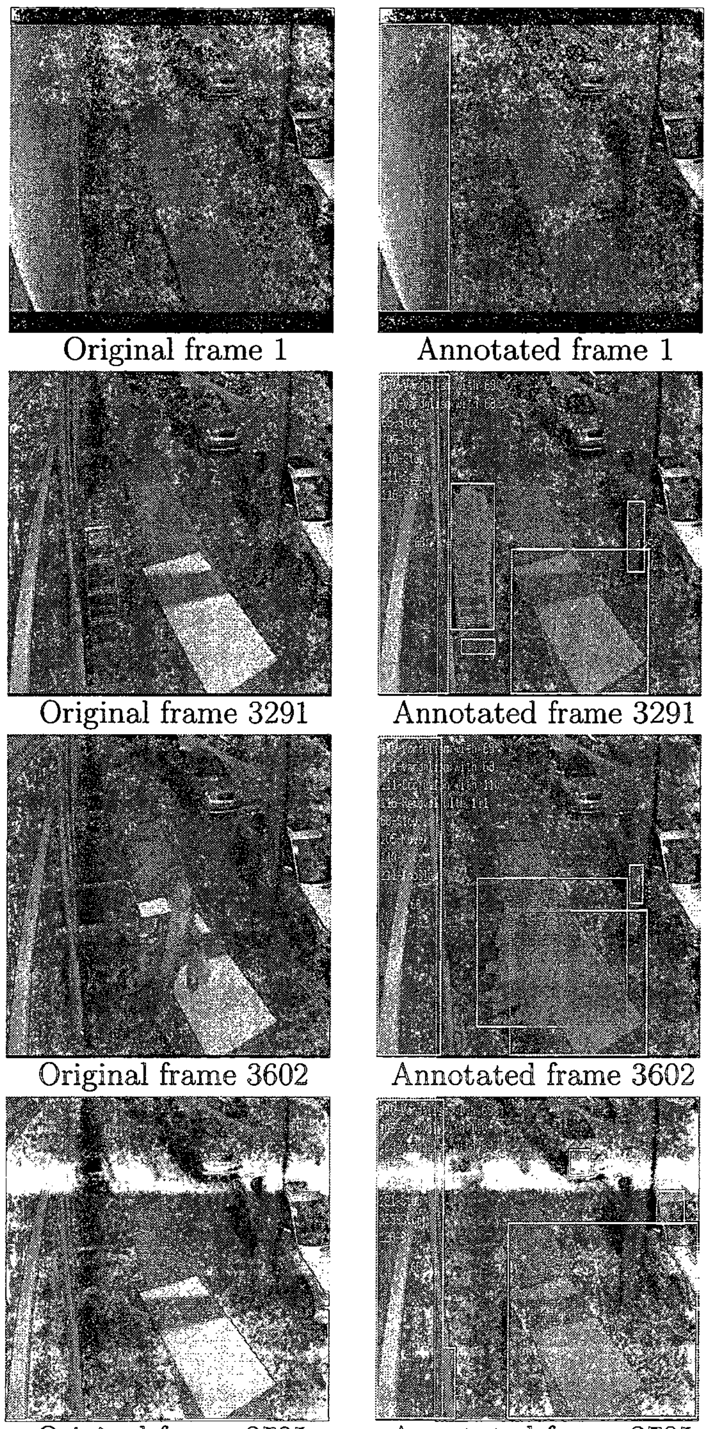
(a) Annotated frame 0 (b) Annotated frame 36

(c) Annotated frame 69 (d) Annotated frame 108

Figure 5.3: Vandalism by spraying over a wall sign. Frame 1 shows the bounding box defining the restricted area in the scene. Annotated frame 108 reads *"0-Vandalism With 2 5"* indicating object 0 (the hand) vandalizing inside the restricted area and the effect showing in objects 2 and 5.

5.4.3 Online Testing

For online testing, we used four vandalism scenarios. In the first scenario, we hanged a poster on a corridor wall monitored by a surveillance camera. Actors were asked to spray paint the poster to simulate graffiti. In the second scenario, a camera filming a locker was used and actors were asked to place a piece of paper (as a simulation of damage) on the locker door. In the third scenario, actors were asked to steal a wireless router from inside a lab monitored by a camera. The third scenario was repeated for a cell phone in a different lab. In the fourth scenario which is intended to test for multiple vandal cases, two actors were asked to simulate inflecting damage to a board in low illumination condition. For the first three scenarios, normal and vandalism interactions were repeated each ten times (four times by two fixed actors and two times by a random passer by). For example, for normal interactions actors were asked to read the poster and leave, take an object from the locker and close it



Original frame 1 — Annotated frame 1

Original frame 3291 — Annotated frame 3291

Original frame 3602 — Annotated frame 3602

Original frame 3705 — Annotated frame 3705

Figure 5.4: Vandalism by removal of a large wall sign in bad illumination conditions. The annotations in frames 3291 read *"110-Vandalism with 68"* and *"111-Vandalism with 68"* suggesting that objects 111 (the man) and 110 (the removed sign) are involved in a vandalism event resulting in object 68 (the damage).

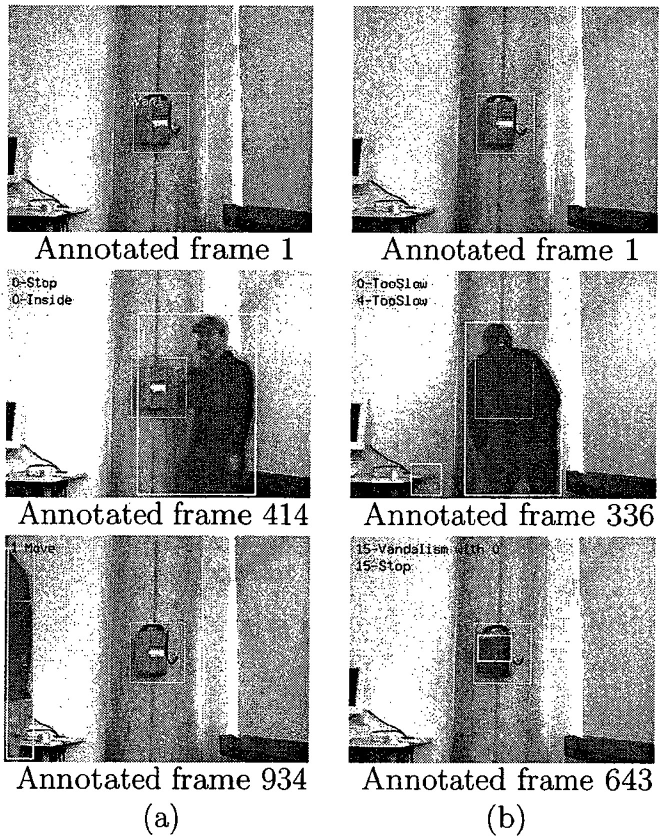
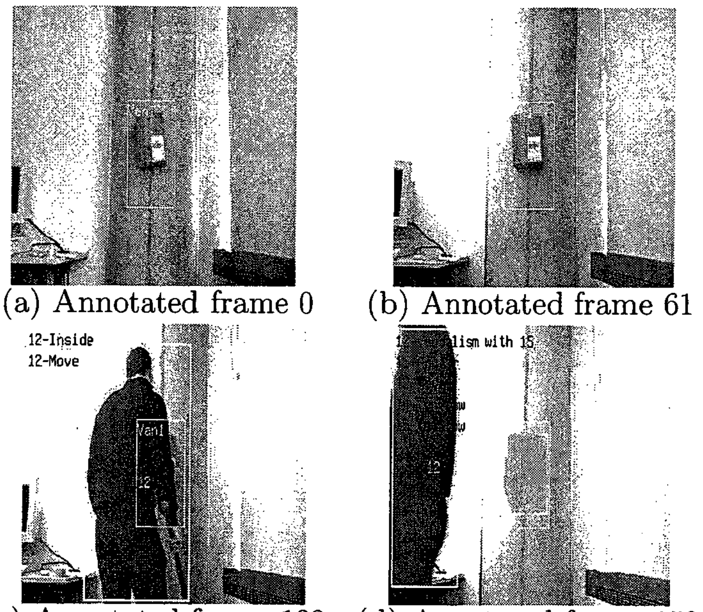


Figure 5.5: (a) Normal interaction with a model pay phone (the actor lefts the handset, simulates a call, replaces the handset and leaves), and (b) vandalism of the pay phone. The annotation in (b) frame 643 reads *"15-Vandalism with 0"* whereas there is no annotation in (a) indicating distinction between normal and vandal acts.

before leaving, and enter then exit the lab without stealing the router. The fourth scenario was repeated five times. In Fig. 5.8 from top to bottom, we show the frames before, during and after vandalism for the graffiti and locker scenarios. Note that Fig. 5.8(a) shows an example in which the overlap between the vandal object and R_r is very small (i.e., vandal marginally enters the site). Nevertheless, vandalism is still detected. In Fig. 5.9(a) we show results for one of the theft scenarios. Fig. 5.9(b) shows the case with multiple vandals. Note that in Fig. 5.9(b), both actors are associated with the vandalism event.

In all normal interactions, vandalism was not detected, i.e., no false alarms. In the two cases where the proposed algorithm did not detect vandalism, the change inside



(a) Annotated frame 0 (b) Annotated frame 61
(c) Annotated frame 128 (d) Annotated frame 150

Figure 5.6: Vandalism of a pay phone with sudden illumination change. Annotated frame 0 shows the first model of the background. The lights are turned on and the sudden illumination changes the background model. Background maintenance provides an updated background model in frame 61 and vandalism is detected in frame 150, which reads *"12-Vandalism with 15"*.

the restricted site was detected moving because of segmentation errors. The proposed method achieves a detection rate of 96%. The detection rate incoporates both true positives and false negatives (i.e., statistical recall). Online and offline results are summarized in Table 5.1.

Table 5.1: Summary of online and offline results.

Scenario	Detection Rate
Graffiti	10/10
Locker damage	10/10
Router theft	9/10
Cellphone theft	9/10
Board damage	5/5
Offline	5/5
Total	48/50

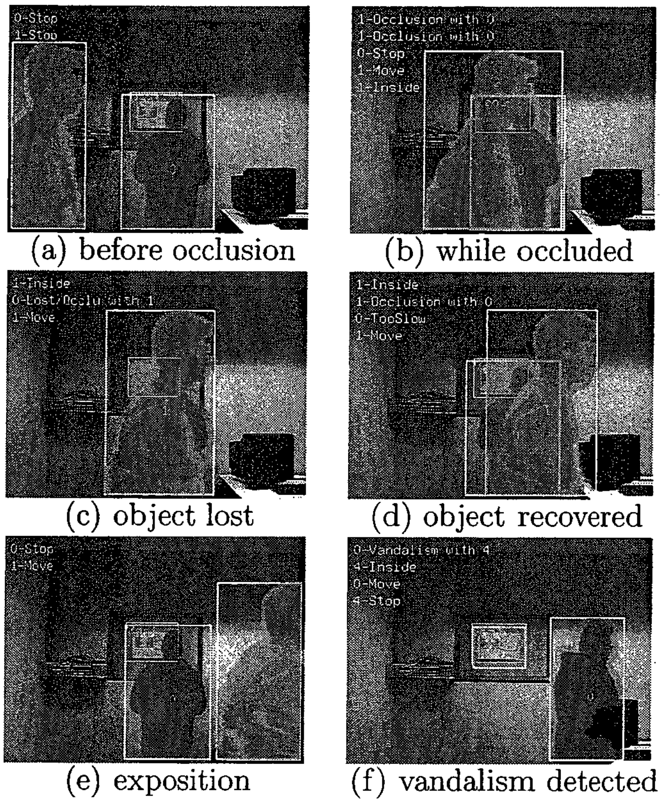
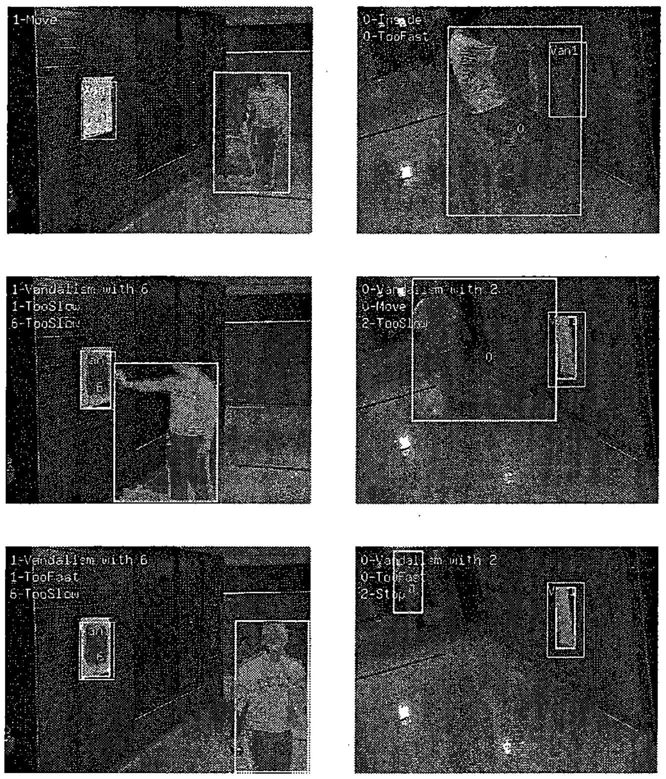


Figure 5.7: Vandalism event detected after complete occlusion. Object 0 (red) is correctly tracked during the occlusion despite being briefly lost.

5.4.4 Vandalism Search

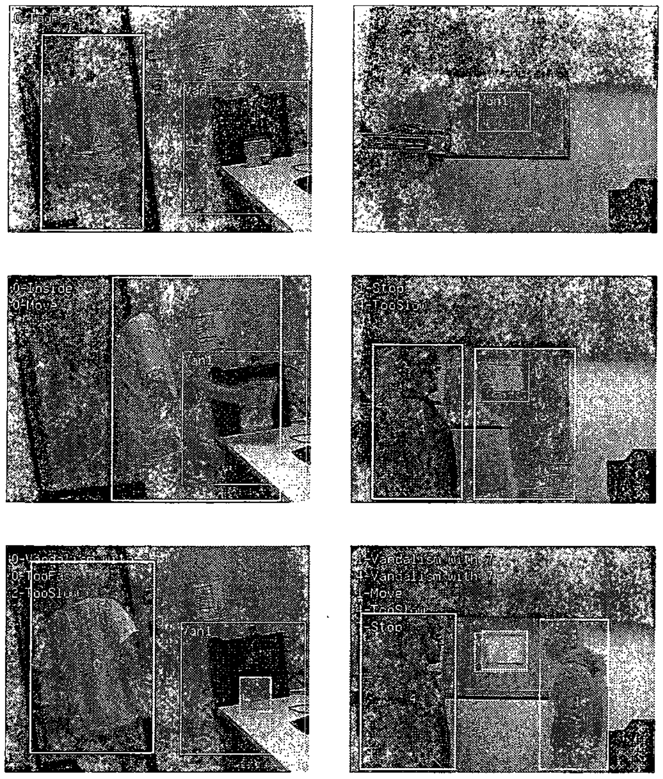
In case it is not possible to detect the vandal such as when a static object stays for too long in the restricted site or when a new object is created due to tracking errors, the proposed method alerts an operator and a search in the event database is performed to identify the vandals, if any. This search is done using the archived surveillance information and related video sequence. Fig. 5.10 shows an example of this search. To the right of Fig. 5.10, we see in (a) an actor enters the scene with ID 0 and spray paints over a poster. During vandalism we purposely force a failure in tracking which leads to loosing object 0 and creating a new object 6 (we can see this new object 6 in the left part of Fig. 5.10). Note that objects 0 and 6 are the same vandal object but they appear to the proposed system now as multiple vandals. In this case, the proposed method is unable to determine if the vandal is object 0 or object 6 and two



(a) Graffiti (b) Damage to Locker

Figure 5.8: Online testing. Part (a) shows the graffiti simulation and the related annotation reads *"1-Vandalism with 6"*. The object does not significantly enter the site and only slightly overlaps with it. Part (b) shows the locker damage scenario and the related annotation reads *"0-Vandalism with 2"*.

events of vandalism are declared separately for each with object 2 being the damage from vandalism as in Fig. 5.10(b). In this case, the multiple vandals are labeled and stored in the database and the human operator can then perform a search to determine the vandal as shown in the left part of Fig. 5.10, thus determining that objects 0 and 6 are in fact the same object. This also shows that the proposed method is robust to such tracking errors.

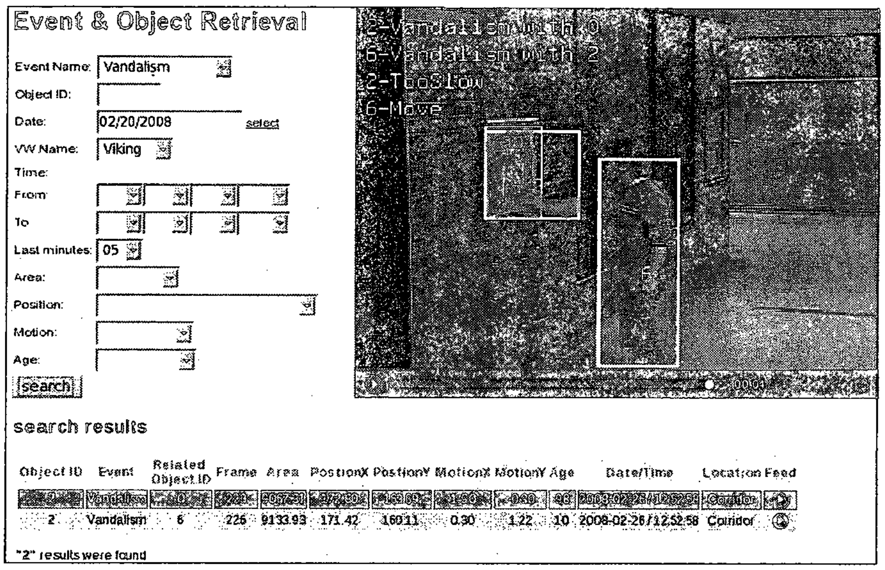


(a) Theft (b) Damage to board

Figure 5.9: Online testing results. Part (a) shows the theft scenario and the related annotation reads *"0-Vandalism with 2"*. Part (b) shows the board damage scenario with multiple vandals detected. The related annotation reads *"1-Vandalism with 7"* and *"4-Vandalism with 7"*

5.5 Summary

This chapter proposed a novel vandalism detection method for video surveillance. Vandalism is detected by monitoring changes inside a vandalism-prone site. The proposed method declares vandalism when an object enters a restricted area in the scene and causes an unauthorized change inside it. When the object leaves the site, we check if the site contains temporally consistent and significant static changes due to damage. Given that the site is unchanged through normal use, we declare vandalism. We tested the proposed method on-line and offline and our results show that it can detect vandalism with a detection rate of 96% while distinguishing between normal



Event & Object Retrieval

Event Name:	Vandalism
Object ID:	
Date:	02/20/2008 select
VW Name:	Viking
Time:	
From	
To	
Last minutes:	05
Area:	
Position:	
Motion:	
Age:	

search

search results

Object ID	Event	Related Object ID	Frame	Area	PostionX	PostionY	MotionX	MotionY	Age	Date/Time	Location	Feed
2	Vandalism	0	220	[illegible]	[illegible]	[illegible]	1.90	-0.40	48	[illegible]	Corridor	
2	Vandalism	6	226	9133.93	171.42	160.11	0.30	1.22	10	2008-02-26/12:52:58	Corridor	

"2" results were found

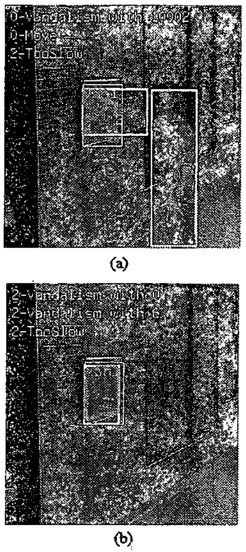


Figure 5.10: Right: (a) an actor (object 0) vandalizes a restricted site and (b) the proposed method declares two vandalism events for objects 0 and 6 with object 2 representing the damage, thus, the vandal is undetermined, i.e., could be 0 or 6. This situation is created by purposely forcing a confusion in the proposed method through losing the track of the original vandal object 0 and making the track reappear as new object 6.
Left: The search for event vandalism in the event database outputs two vandalism events or two vandals, object ID 0 and object ID 6. The human operator can then view the related video sequence for verification of details.

and vandal behaviors. The proposed method integrates efficiently in online video surveillance systems without significant increase in computations.

Chapter 6

Scalable and Distributed Surveillance System

6.1 Introduction

Most surveillance systems presented in the literature focus on the video-processing algorithms and marginally consider the architecture and challenges of building an end-to-end surveillance system. This chapter proposes a scalable, distributed, and real-time video-surveillance system with a novel system architecture, and indexing and retrieval capabilities. We use indexing to refer to the process of extracting, delivering, and archiving surveillance information in a database for searching. The chapter also addresses the challenges faced in building this system, e.g., how to distribute the processing to achieve a high frame rate and how to index surveillance information to reduce storage requirements. The system consists of three modules: video workstations for signal processing and indexing, web-based control workstations for monitoring and control, and a server for overall system management, archiving and retrieval. Video workstations capture, process, encode, and stream surveillance video and extracted surveillance information over the network. Control workstations display received surveillance feeds alongside the information extracted from them. They also provide, through the server, access to archived object and event based surveillance video and information. The proposed system reduces storage requirements by modeling object features as temporal Gaussians and uses variations in object features to determine the temporal separation between the Gaussians. Our results show that the proposed system yields real-time video feeds of 18 frames/second for SIF video, reduced network and storage usage, and precise retrieval results. Results also show that the proposed system architecture produces more scalability and balanced distributed performance than recent surveillance system architectures.

6.2 Review of Related Work

Due to increased security concerns, automated real-time surveillance systems receive considerable attention from academia [91, 104–106] and the industry [107]. Utilizing such surveillance systems is now a necessity in airports, subways, offices and even homes to aid in detecting events of interest or potential security risks [108, 109]. In general, desired features of video-surveillance systems are: 1) scalable architecture; 2) affordable hardware requirements; 3) real-time performance; and 4) reliable alerting mechanisms. Designing automatic surveillance systems to meet these requirements promises a significant increase in their deployability [110].

Recent surveillance systems in the literature can be categorized into specialized [111–115] and generalized systems [116–119]. The specialized systems in [111–115] are built for train or railway surveillance [111], traffic or highway surveillance [112, 113], elevator surveillance [114], and home surveillance [115]. These systems are tuned to specific application needs, hence are less useful for others.

Generalized systems are further categorized into centralized [116, 117] and distributed systems [118, 119]. Centralized systems deliver lower performance due to the high computational burden on the central module [120]. These systems focus on the video-processing aspect of the system without specific attention to the overall surveillance system architecture, which leads to unbalanced system load and decreased scalability. Alternatively, distributed systems [9, 118, 119, 121] make use of communication protocols to divide the work amongst a network of less powerful computers, thus balancing the overall system load and increasing scalability. Also, distributed systems facilitate the implementation of complex video-processing algorithms on existing hardware resources because of decreased load on the video-processing modules (e.g., by delegating management, monitoring and control to other modules). Some distributed architectures, however, can still benefit from further load-balancing features. For example, the systems in [118] and [119] do not separate between the monitoring, control, management, and archiving modules. Their architectures consists of a set of video-processing modules connected to a single server. All surveillance videos are displayed at the server side leaving only one point where surveillance information is accessible. This reduces scalability which is key for practical systems [110]. In the work presented in [121], the authors proposed a surveillance system architecture in which a video processing module streams information to a video monitoring module. A management server controls permission to view the streams and a media server is used to store extracted information. In the surveillance architecture presented in [9], the video processing module is connected directly to a storage server which relays compressed video sequences to a transmitter to which viewing and control units are connected. Both systems in [9, 121] can benefit from increased network bandwidth savings and less processing delays through further task decomposition.

In this chapter, we propose a scalable, distributed and real-time surveillance system with indexing and retrieval of surveillance information. The proposed system can be used to realize many surveillance applications. The advantages of the proposed surveillance system include: 1) a novel and scalable system architecture with extensible, and reliable distributed modules; 2) real-time indexing and retrieval of object- and event-based surveillance information and video; and 3) reduced network and storage requirements through reduction of features sampled over time.

The remainder of the chapter is organized as follows: Section 6.3 presents the proposed system architecture. Sections 6.4-6.6 present the proposed system modules: video workstations, server, and control workstations, respectively. Section 6.7 presents the results, and Section 6.8 summarizes the chapter.

6.3 Proposed Surveillance System Architecture

The proposed system architecture consists of three independent and distributed modules: *video workstation*, *control workstation*, and *surveillance server.* These modules and their connections are depicted in Fig. 6.1.

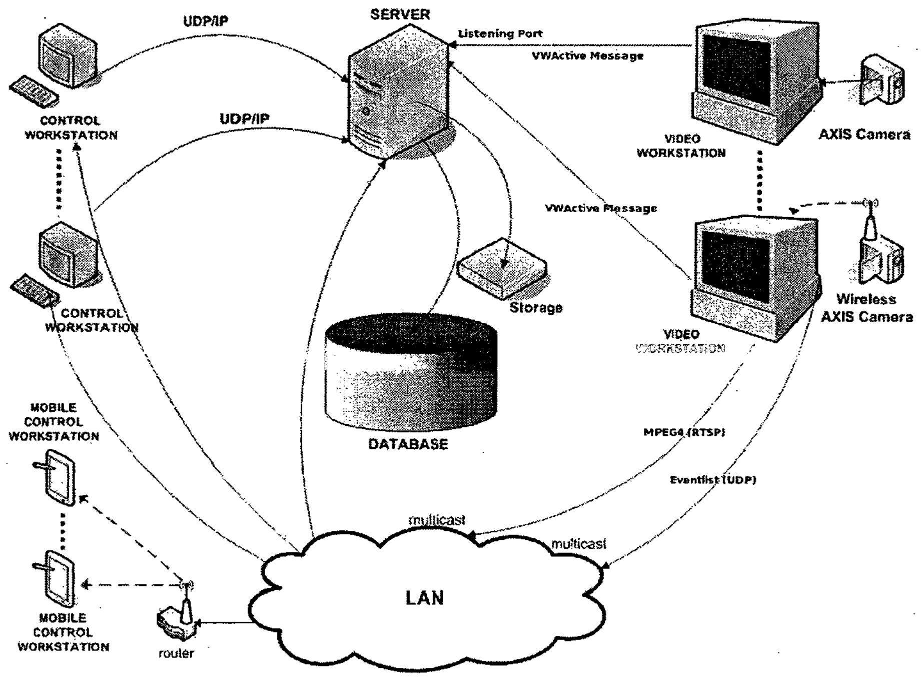


Figure 6.1: Architecture of the proposed video surveillance system.

A video workstation captures the MJPEG video signal from one or more cameras and processes (e.g., extracts objects and events) the decoded signal. It further

MPEG-4 compresses the video signal, and streams it to a multicast network address using the real-time streaming protocol (RTSP). At the same time, extracted surveillance information (objects and events) from each frame is encapsulated in a user datagram protocol (UDP) packet and streamed to a different but associated multicast address. Throughout this chapter, we use the term *surveillance information* to refer to extracted video object features and detected video events (e.g, the area of an object or the event an object has vandalized a restricted site).

The server archives, using a database, all communication between video and control workstations. For example, it archives surveillance information by joining the group of control workstations listening to the mutlicast addresses to where video workstations stream surveillance information. Similarly, it stores as MPEG-4 elementary streams the surveillance video sequences by joining the multicast addresses to where video workstations are streaming processed surveillance feeds. Other tasks of the server include maintaining security information about the access privileges of all control workstations and providing control workstations access to the surveillance database. A control workstation operator could query this database for object, event, or combined object and event video sequences or information. Both extracted surveillance information and related video sequences are then returned to the operator.

A control workstation views surveillance feeds and issues commands to one or more video workstations. It is designed as a web-based application and utilizes standard embedded media players to view the RTSP surveillance-video streams. Commands are created in response to graphical user interface (GUI) events (e.g., clicking on the GUI buttons) and are sent to the server and control workstations as hypertext transfer protocol (HTTP) requests or TCP packets. We program the control workstations to respond to receiving high-level surveillance information using a set of alerting mechanisms. With the control workstation being a web-based application and the surveillance stream being MPEG-4 compressed, we provide access to both the live and archived surveillance video sequences from mobile devices.

Each one of the system modules: video workstations, control workstations, and surveillance server, is built from a set of modularly designed functional components (see Fig. 6.2). A video workstation is denoted by V_i, a control workstation by O_j, and the surveillance server by S. The design of the system architecture is influenced

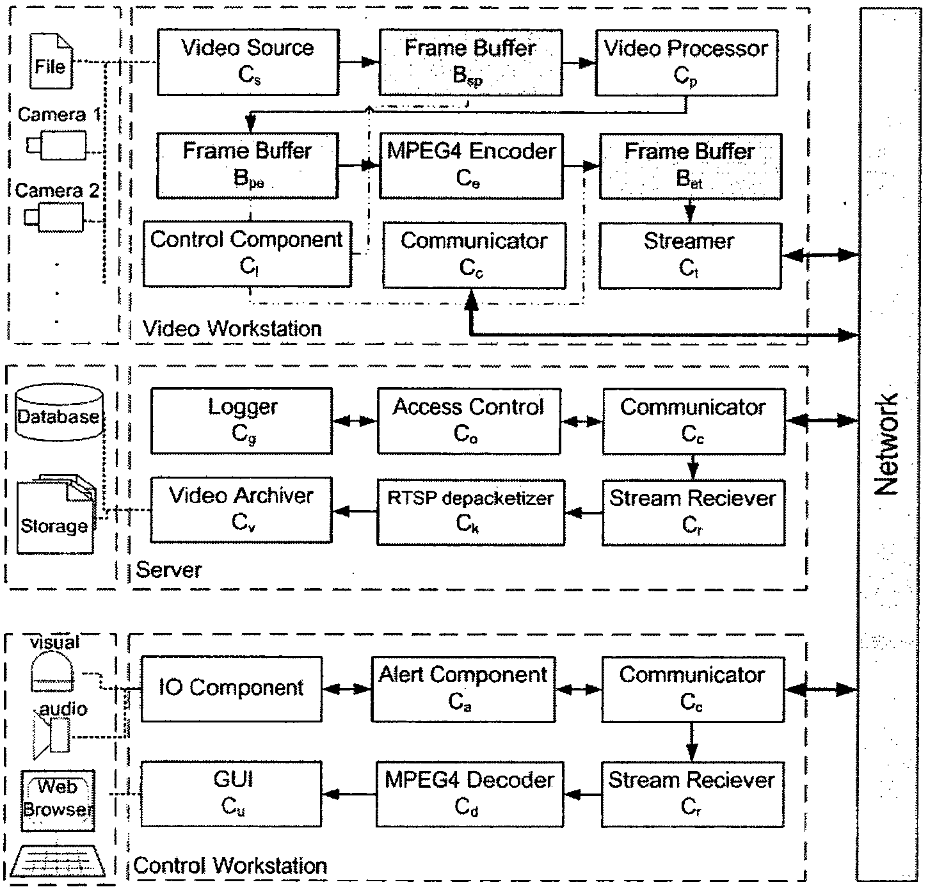


Figure 6.2: Block diagram of system modules and their components.

by the steps used in video processing. The main principle is to keep VW focused on processing by delegating as many of its responsibility as possible to other modules. For example, this has motivated the use of distributed architecture and a tri-module system, parallel processing in VW, and the use of multicast to stream surveillance video and information only once.

6.4 Video Workstation

Each instance of the video-workstation module, V_i, is built as a set of components and buffers, i.e.,

$$V_i = \{C_x^{V_i}, B_{xy}^{V_i} | x, y \in \{s, p, e, l, c, t\}, x \neq y, xy = yx\}, \tag{6.1}$$

where $C_x^{V_i}$ stands for a component in the i^{th} video workstation, $B_{xy}^{V_i}$ for a buffer between components x and y (or y and x) in the i^{th} video workstation, s for a video source, p for processing, e for encoding, l for control, c for communication, and t for streaming (see Fig. 6.2). C_s captures the video signal and delivers it to B_{sp}. C_p grabs frames from B_{sp} and processes them to label moving objects and extract surveillance information. C_p writes processed frames to B_{pe} for C_e to MPEG4-encode them. C_t streams encoded frames to a multicast address. C_t also streams surveillance information as UDP packets to an adjacent multicast address.

6.4.1 Video Processing and Indexing

V_i processes s using video-processing algorithms such as object segmentation and tracking or object classification. It then creates a video index of source s. In this chapter, the surveillance video index, which we also call surveillance information, consists of the extracted objects with their features and events with their timing information (e.g., frame 310 is where object 1 vandalized object 2 or frames 200, 2341, and 9870 are where large objects such as buses appear in the scene). This index enables us to search and locate parts of the surveillance video with interest to surveillance such as footage of a vandalism event or footage of object 3 during its life-time in the scene. The video-processing algorithms required in this chapter are shown in the UML sequence diagram of Fig. 6.3. In the first *motion detection* step, we use the new frame and the background frame model to classify pixels into moving or foreground pixels and static or background pixels. Then, we use the binary frame produced for *object detection*, in which we extract spatio-temporal features from blobs converting them into *image objects.* More details on segmentation is found in Chapter 3. Next, we pass the image objects to a feature-based object tracker in the *object tracking* step, which builds correspondences between them in consecutive frames creating tracks or *video objects.* More details on how tracking is performed is in Chapter 4. Finally, we use the tracking information to detect events such as vandalism in the *event detection* step. More details on how we detect vandalism is in Chapter 5. Any algorithm for motion detection, object detection, object tracking, and event detection can be used. Obviously, the performance, in terms of frame rate, accuracy, and storage, of the proposed system depends on the used video-processing algorithms. Fig. 6.4 visualizes the output of each step of the required video-processing algorithms.

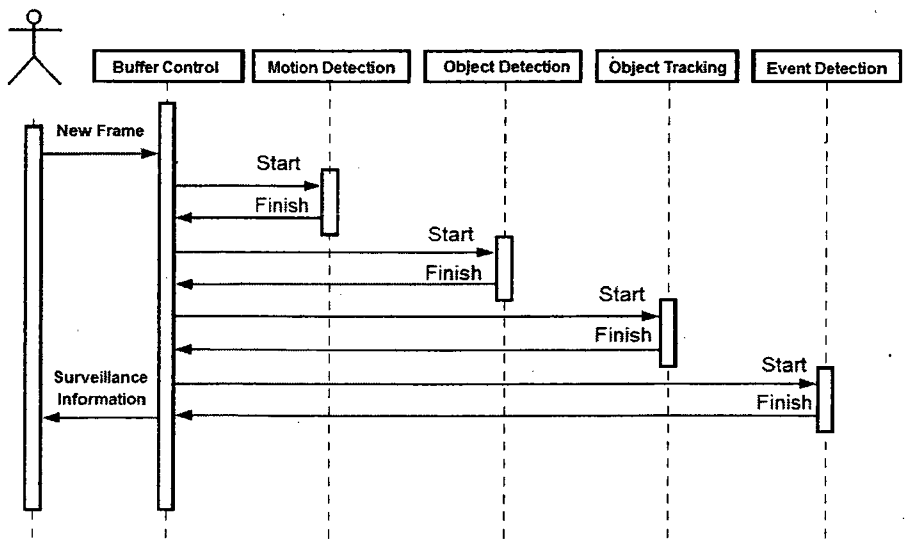
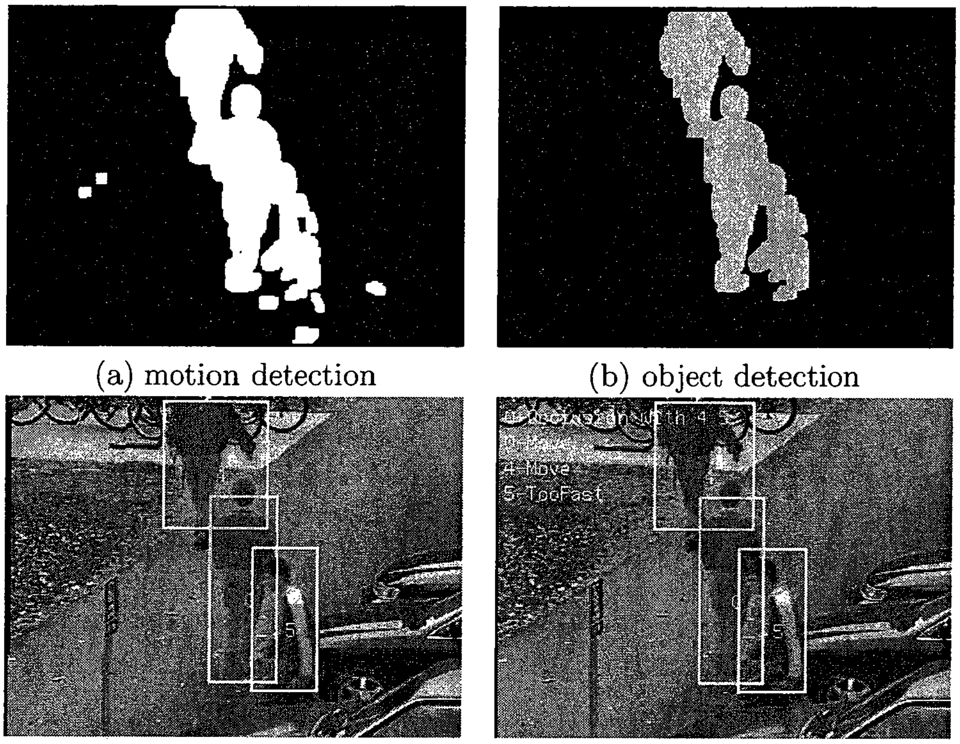


Figure 6.3: UML sequence diagram showing processing steps used to index surveillance video signals.



(a) motion detection (b) object detection

(c) object tracking (d) event detection (annotated)

Figure 6.4: Example output of each step of the video processing required. Note events are overlayed on the top left corner of (d).

6.4.2 UML Design of the Multi-threaded Video Workstation

Fig. 6.5 presents the design of the video workstation components and buffers with a UML class diagram. All components inherit from class *Filter* which inherits from class *Thread* to create a multi-threaded framework. Multi-threading is key to enable a video workstation to produce fluid video despite significant delays from video processing and compression. Because buffers are shared resources between two components, they inherit from class *Mutex*, which is an operating-system lock to organize access to the shared resource and ensure the components architecture are time-safe. The proposed buffers are essentially queues with synchronized access. Every *Filter*, or component, has two buffers connected to it. The first buffer is an input buffer which connects it to the previous functional block. The second buffer is an output buffer which connects it to the next functional block. Each buffer is aware

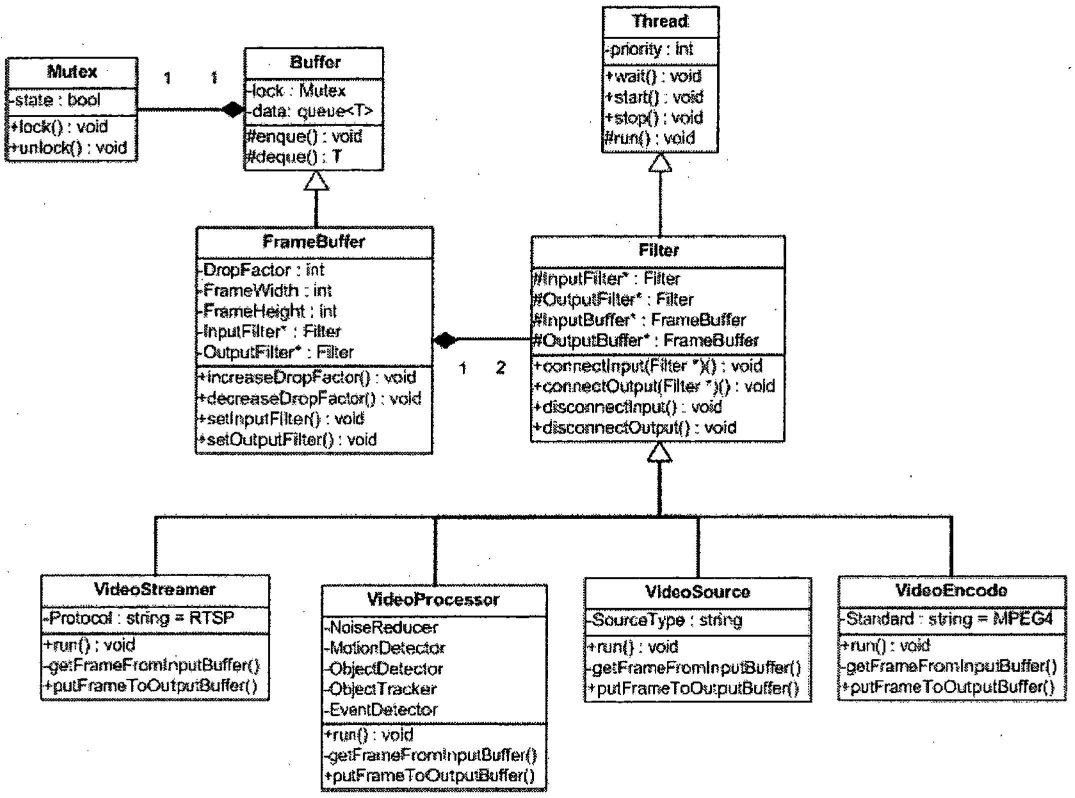


Figure 6.5: UML design of a video-workstation components and buffers. All buffers are synchronized queues. All components are filters with two buffers for input and output. Buffers are aware of their input and output components. The *VideoProcessor* is where one or more video-processing algorithms are realized, e.g., object segmentation or event detection.

of the two filters connected to it. There is no input buffer for the video-source (C_s) component and no output buffer for the video-streaming component (C_t). We extend class *Buffer* one more level down the inheritance hierarchy to produce *FrameBuffer* which hosts objects that represent video frames.

The *VideoSource* class has a generic interface. We extend it to create classes which define a particular video source. The extension includes specific routines and variables necessary to access the video source. The *VideoStreamer* is an interface for streaming classes. In our realization, we extend the *VideoStreamer* class to produce the *RTSPStreamer* class which will create a multicast group for control workstations and the surveillance server to join. The *VideoEncoder* class is an interface to compression classes. We extend it once to create the *MPEG4Encoder* class. The core class of the proposed system is the *VideoProcessor* class. Extending this class creates the system video processing engine which defines the overall surveillance functionality of the system.

The proposed architecture is designed to account for the failure of one or more components. As can be seen from Fig. 6.5, components have generic interfaces which allow for real-time component switching in the case of component failure or in response to operator requests. For example, a number of key components which are prone to failure (e.g., processing component) are started and kept in a suspended state during the normal operation of the system. If the main processing component fails, the thread-scheduling component halts the system until a replacement is integrated.

6.5 Surveillance Server

A video workstation sends the multicast addresses for the surveillance video signal and information as part of its registration with the surveillance server. The server's communicator receives this information (see Fig. 6.2) and stores it in the database. When a control workstation registers with the server, it receives a list of connected video workstations $\{V_i\}$.

The server is not the central point in the proposed system, this is different from traditional surveillance system architectures [9], where the server is central and represents a single point of failure in the system. A failure in the server does not cause any of the video workstations to lose the surveillance feeds. The server joins (i.e., becomes another receiver) the multicast addresses to which all video workstations stream video signals and extracted information, and is hence another receiver in the system specifically tasked with archiving. Therefore, multiple servers can be added to scale the system. The server stores surveillance video signals in the storage media and extracted information in the database. The stream receiver C_r receives the video, the RTSP depacketizer C_k depacketizes it and the video archiver C_v archives it in the database and storage media. The communicator C_c receives surveillance information and the logger C_g stores it in the database.

6.5.1 Delivery of Surveillance Information

We use the surveillance protocol in Table 6.1 for inter-module communication (e.g., surveillance events from video to control workstations and control commands from control to video workstations). The communication takes place between communication components $C_c^{V_i}$, $C_c^{O_j}$ and C_c^S (see Fig. 6.2 and equation (6.1)).

Table 6.1: Structure and examples of surveillance protocol packets.

Message Type	Size	Source	Destination	Message
ObjectList	1KB	$C_t^{V_i}$	Multicast IP	Obj1, Obj2, ... etc
EventList	500B	$C_t^{V_i}$	Multicast IP	Ev1, Ev2, ... etc
Command	100B	$C_c^{O_i}$	$C_c^{V_j}$ & C_c^S	Move Camera

We send all commands to both the server for logging and the destination video workstation. There are two types of commands. The first is sent directly from the control workstation to the cameras such as the PTZ controls as HTTP requests. Other commands related to video surveillance are sent to the video workstation as TCP packets and handled there (e.g., the command to change the quality level of the surveillance video or change how objects are labeled for improved visualization).

6.5.2 Modeling feature variations as temporal Gaussians

One of the main challenges in a surveillance system is managing a large volume of extracted surveillance information, here object and event data. The surveillance server receives up to 18 frames per second of surveillance SIF video. With event and object data being received every frame from multiple cameras, we address the problem of archiving this information. Event information, such as when the event started and what are the related objects, can be filtered based on the event type. Events that repeat very often like *Move* or *Stop* are less likely to be of interest, whereas major events such as *Vandalism* or *Stay-Long* are often of interest. Therefore, the main challenge of archiving surveillance information is with object features, which are present every frame. Moreover, when performing a query involving both objects and events, objects features exhibited during the life cycle of the event must be available.

The problem of archiving (or sampling) frequent object information can be considered as an object feature modeling problem. The simplest model is to use all extracted features of the object in all frames. However, this approach, while it provides the most accurate object information for any object-event query, keeps the system busy and has a high storage requirement. In the proposed system, we model object features as a temporal sequence of Gaussians. Let $T = P \times k$ for $k \in \{0, 1, 2, ...\}$ be the index of the temporal Gaussians sequence, i.e., $T \in \{0, P, 2P, ...\}$, and k is the frame number. P controls the temporal separation between the Gaussians. The mean μ_T^k and variance v_T^k of one Gaussian in the sequence is calculated from incoming feature observations using

$$\mu_T^k = \rho \mu_T^{k-1} + (1-\rho) A^k, \tag{6.2}$$

$$v_T^k = \rho v_T^{k-1} + (1-\rho)(A^k - \mu_T^k)^2, \tag{6.3}$$

where A^k denotes an example object feature in frame k (e.g., area), $\rho = (N-1)/N$ is a scaling factor that is a function of N, and N is the current number of observations in μ_T^k and v_T^k. When $N = P$, new μ_T^k and v_T^k are created and N is reset to 1. Example features include area, perimeter, position, and minimum bounding box (MBB). They are maintained and used to calculate more complex features such as contour irregularity. The variance of the temporal Gaussian v_T^k measures μ_T^k's reliability as a representative of features over the period P. Small values of v_T^k means the object features remained stable over time possibly because the object stopped moving in the scene or started moving slowly (e.g., due to traffic light). v_T^k can be normalized to create a probability indicator of feature reliability that is returned with μ_T^k in the retrieval results.

Selecting a proper value of P depends on the frequency of feature variations. For example, a car moving perpendicular to the camera has small feature variations compared to a non-rigid object such as a pedestrian moving towards the camera. We set $P = 10$, We motivate our choice for $P = 10$ from the Nyquist sampling theorem based on the maximum observed object feature frequency in typical surveillance video sequences. We can relate the variations of object features over time to one of three reasons: 1) a gradual increase or decrease in some object features as it moves towards or away from the camera; 2) a periodic variation of features due to the movement of arms and legs while walking or running; and 3) random variations due to noise from local illumination changes or segmentation errors. We can view the gradual variation of object features as periodic if we consider the movement towards or away from the camera as part of an imaginary cycle in which the object performs the opposite all the time (e.g., move towards the camera then move away repeatedly). The variations due to body motion is largest when the object moves perpendicular to the camera. We can have a combination of variations due to gradual change and periodic motion, but the periodic motion causes the change with the highest frequency.

We can observe the periodic-motion high frequency in Fig. 6.6 showing object binary images taken from the reference test sequences *Comm2* (top) and *Ekrlb* (bottom). Also, Fig. 6.6 shows plots of object area over time from sequences *Comm2* (left) and *Ekrlb* (right). In *Comm2*, the object moves outdoors far from the camera and the video is captured at 60 frames/second. In *Ekrlb*, the object move indoors close to the camera at a frame rate of 30 frames/second. Note the semi-sinusoidal pattern in the object area plots. We consider two cycles in this semi-sinusoid to represent one period of object movement because the full visual content of an object is revealed from the camera's point of view after two walking steps to account for asymmetrical situations (e.g., when an object is holding a bag in one hand). From Fig. 6.6, the periods of the maximum frequency are $\frac{180-115}{60} = 1.08s$ and $\frac{88-57}{30} = 1.03s$, respectively. Accordingly, Nyquist theorem tells us to sample at a period of $0.5s$. We set $P = 10$ which corresponds to sampling periods $0.16s$ and $0.33s$, respectively, which is below the Nyquist sampling period of 0.5s. With the proposed sampling scheme, the storage requirement is significantly reduced. This, however, creates a delay of one third of a second for 30 frames per second video sequences between the object features and the event. This delay is insignificant given the normal speed of objects in surveillance sequences.

6.5.3 Surveillance Database Design

In the proposed system, surveillance information is archived in a database at the server, while surveillance video is stored as an MPEG4 elementary stream in a storage unit connected to the server. The part of the database design related to surveillance information and video is shown in Fig. 6.7. Database tables related to access control, logging, and other management tasks are not shown due to space limitations. Every *VideoWorkstation* has one or more *SurveillanceVideos.* Events and objects each span two tables: the first is *VideoObject/EventType* for video-wise (or global) information and the other is *VideoObjectFeatures/Events* for frame-wise (or local) information. The two tables are linked by their IDs. Each row of the *VideoObjectFeatures* table represents the average (see equation (6.2)) of features over P frames. Complex queries involving both objects and events require linking their tables (e.g., when trying to find the features of an object with a particular event). Since object features are not

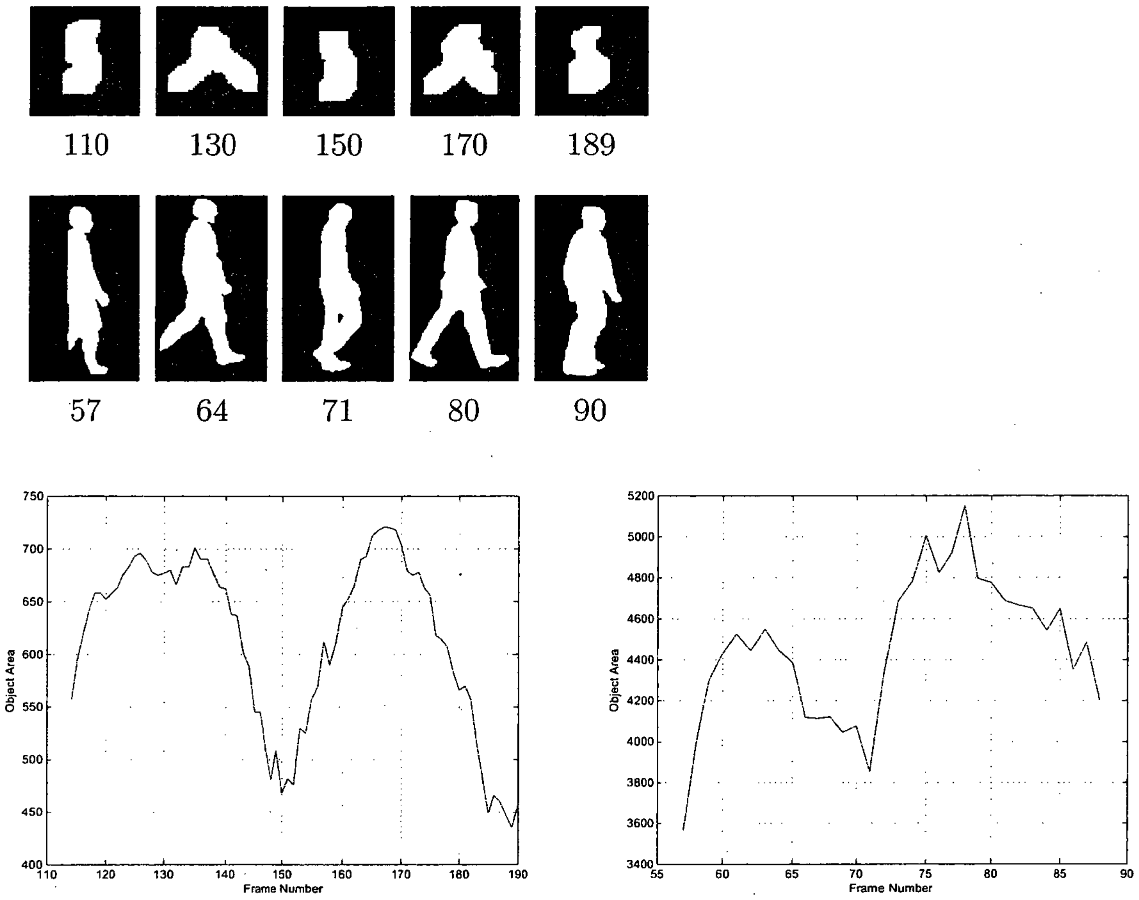


Figure 6.6: Change of object area (as a representative of object features) due to periodic motion over time for test sequences *Comm2* (left) and *ekrlb* (right).

present for every frame due to sampling using equation (6.2), we return features which minimize the absolute difference between the object and event frame numbers

$$R = \{Obj, Ev: \quad \min |k_{Ev} - k_{Obj_T}|\}, \tag{6.4}$$

where R is the set of returned results, Ev and Obj are the query event and the object features, respectively, and k_{Ev} and k_{Obj_T} are the event and object frame numbers, respectively. We use the archived event and object timing information to locate, extract, and return the parts of the surveillance video associated with the object or event query.

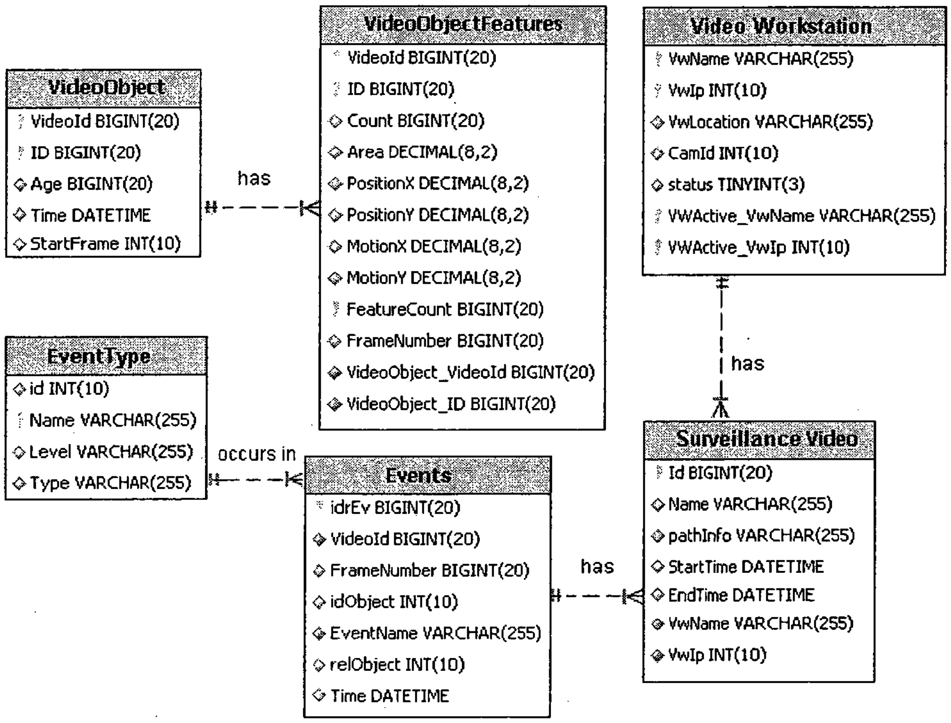


Figure 6.7: Schema of surveillance system database at the server.

6.6 Control Workstation

Control workstations provide the main form of user interaction in the proposed system architecture. The main components of the control workstation are: 1) a communicator C_c to receive surveillance information; 2) an alert component C_a to filter surveillance information and respond with pre-programmed alerts; and 3) a pipeline for viewing the video signal consisting of a stream receiver C_r, an MPEG4 decoder C_d, and a GUI C_u. C_u also receives control commands from the user.

6.6.1 Presentation of Real-time Surveillance Information

We present real-time surveillance information in three panels as shown in Fig. 6.8. The first panel (top left) shows the real-time surveillance video. Objects (and their IDs) are optionally shown either inside their MBB or their color is changed (see an example in Fig. 6.15). Events are shown in the second panel (top right) and

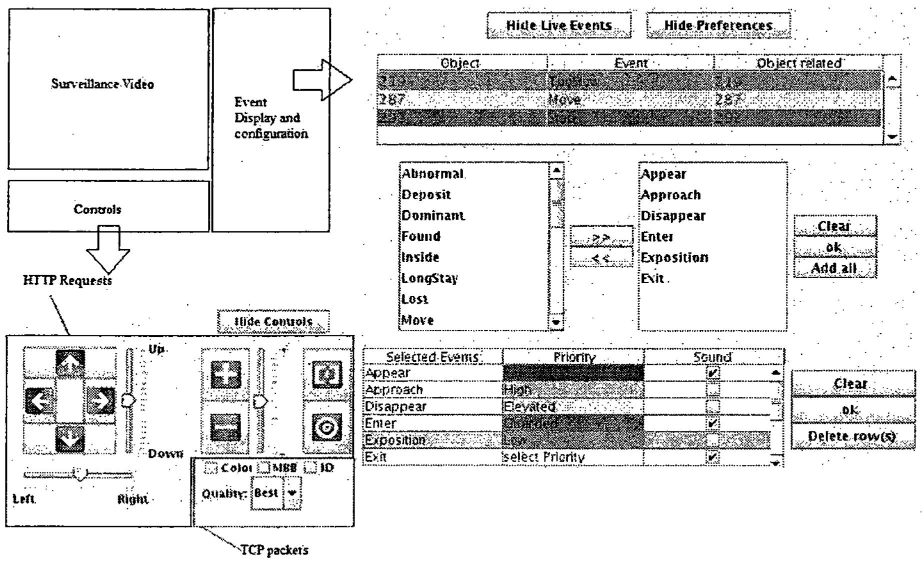


Figure 6.8: Presentation of real-time surveillance information and video signal.

optionally overlaid on the top left corner of the surveillance video. This allows an operator to focus entirely on the surveillance video while seeing the full information. Alternatively, an operator may configure the event list to allocate a certain color and audio signal to a particular event of interest (e.g., vandalism), which draws the operator's attention to the surveillance video quickly. Controls are placed in the final panel below the surveillance feed. The server enables only controls for which the user has access.

6.6.2 Retrieval of Surveillance Information and Video

The user can retrieve events using their type, camera, or timing information. We retrieve objects using their camera and timing information and based on a criteria that puts object features in their context. The following list shows the used search criteria for objects and its relattion to stored features:

- Area: small, medium, and large.
- Position: right, left, center, top, bottom, .. etc.

- Motion: slow, fast, and too fast.
- Age: short, long, and very long.

The criteria allow us to translate object features and events into contextual queries. For example, using the area and position features, we can search for buses (large objects) exiting a street (bottom left) of the scene. Another example is searching for cars (medium objects) which violate the speed limit (too fast) or have parked in a location for a long time (very long age). The specific limits used in the above criteria, or other criteria, can be designed from specific application needs.

6.7 Results

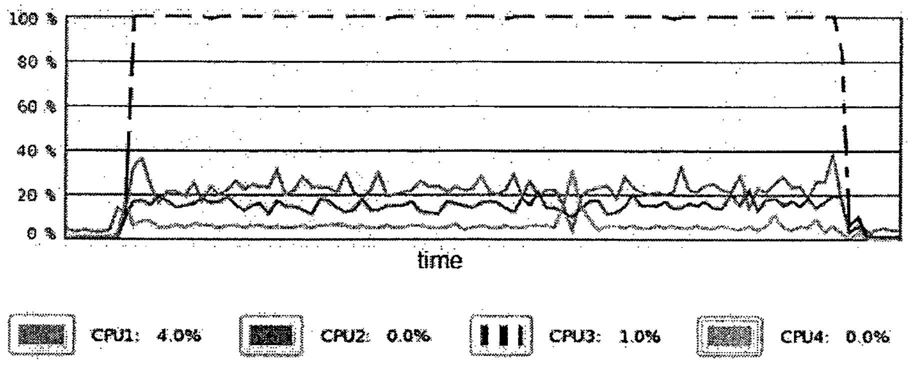
We test the performance of the proposed surveillance system by focusing on the performance of indexing, retrieval, delivery, and presentation. We later test the performance of segmentation, tracking, and vandalism detection in their Chapters 4, 5, and 6, respectively. We used the event detection algorithm in [122] to extract other events. Other algorithms can also be used (e.g., [123]). We first examine the resource (CPU, network, and memory) requirement of video workstations and the factors which affect the final frame rate in Section 6.7.1. Then, we compare the proposed system architecture to a recently presented architecture in terms of scalability and distributed performance in Section 6.7.2. Finally, we present the results of testing the indexing and retrieval capabilities of the proposed system in Section 6.7.3. To test the proposed system online, we used the network camera models AXIS213 and AXIS233D with a 1/4" CCD sensor set to produce 30 frames/second of progressive SIF video.

6.7.1 Resource Requirements and frame rate

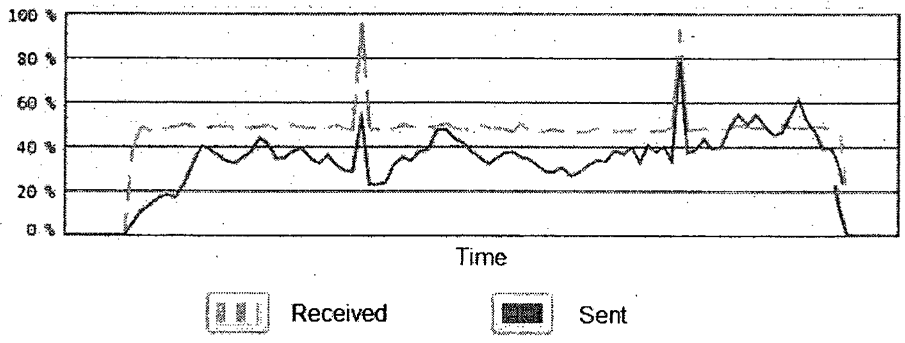
Video workstations are the resources intensive modules of the system. We tested their resources utilization over time. The results are shown in Fig. 6.9. In this test, we ran the system for 15 minutes while monitoring busy morning traffic. We noticed that the amount of memory used is stable over time. CPU usage over time is shown in Fig. 6.9(a). The video workstation needs 33% of the processing power (the full CPU3 in 6.9(a)) of a quad-core computer to produce an 18 frames-per-second surveillance SIF video. A possible lower bound on the frame rate is 10 frames/second. Lower than 10 frames/second leads to non-fluid surveillance video at the CW side and many frame drops, which lead to less reliable tracking. The video workstation network utilization is shown in Fig. 6.9(b). We can observe a steady MJPEG input stream coming from the camera and a variable output MPEG-4 and surveillance information streams depending on the complexity of the scene. As can be seen, resources usage is stable over time.

To test the effect of video content on the frame rate, we use video signals with variable number of objects and occlusions in the scene. We notice no change in the final frame rate as more objects are introduced as in Fig. 6.10. We observed temporary drops in the frame rate when the complexity of the scene increases as measured by the number of occlusions. The frame rate of the system increases again when the occlusions end. This expected drop in the frame rate is necessary to achieve reliable indexing.

The time needed to process one frame in the video workstation can be divided into three parts associated with components C_s (capture), C_p (processing), and C_t (streaming). The breakdown of their relative time is shown in Fig. 6.11. Within processing, the ranking of video processing tasks based on their effect on the frame rate of the system is as follows: 1) occlusion and split detection and correction; 2)



(a) VW CPU utilization over time



(b) Network utilization over time

Figure 6.9: Resource (CPU and network) utilization over time for video workstations. Memory usage was stable over time and is not shown.

segmentation; 3) tracking without occlusions and splits; and 4) event detection. We draw two conclusions from Fig. 6.11: 1) to increase the output frame rate of the

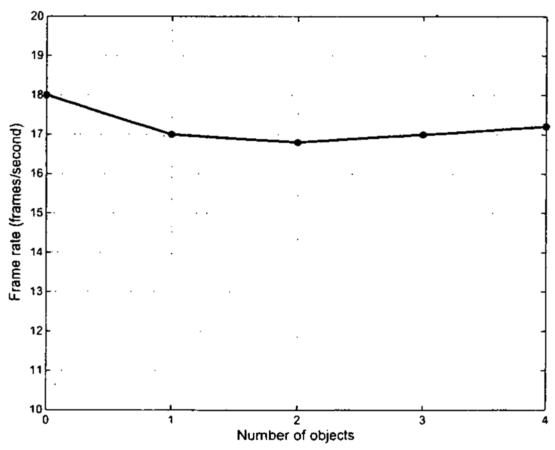


Figure 6.10: Frame rate under different object counts.

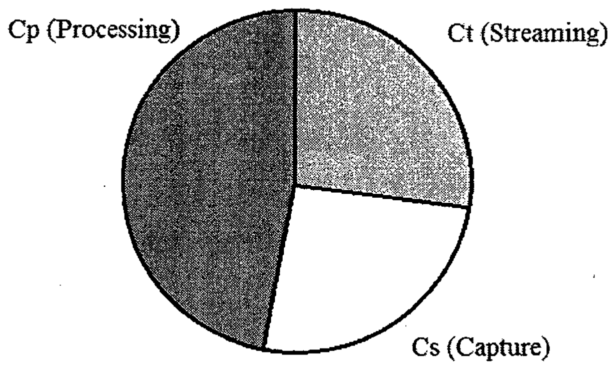


Figure 6.11: Percentage of time used in the three major components of the video workstation to process one frame. Note that the C_s component includes MJPEG decoding and the C_t component includes MPEG4 decoding.

Table 6.2: Frame rate using different CPU Types

CPU Type	# Cores	Core Speed (MHz)	Frame Rate
Intel Core i7 920	4	2670	25
Intel Xeon	4	3200	18
AMD Opteron 248	2	2192	10

system, we aim to have the time needed for C_p to match that of C_t or C_s; 2) we can improve the frame rate of the system running on a multiple core computer by associating C_p with one CPU core and C_s and C_t with another. We can use the second conclusion to establish that the ideal number of video workstations running on the same computer is half the number of CPU cores in that computer. We test this result in Fig. 6.12 by running one, two, three, and four video workstations consecutively on the same quad core computer. With the proposed component scheduling scheme, we do not detect frame rate drops for the first two video workstations. The frame rate starts to drops when running three video workstations at the same time.

We tested the proposed system on three different CPUs and conclude that the type of CPU affects the frame rate as can be seen from Table 6.2. Note that we used the Intel Xeon processor (row two), C++, and Linux in all other tests of the system.

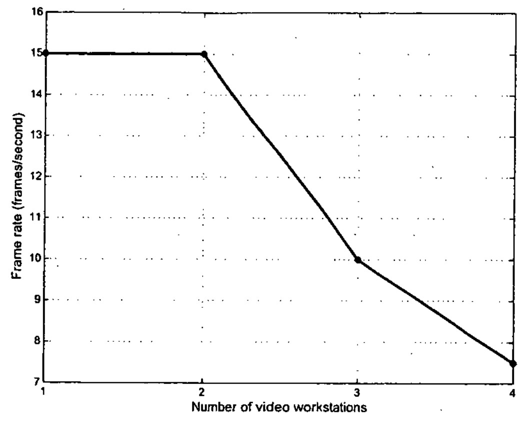


Figure 6.12: Effect of running multiple video workstations on the same quad core computer by associating the C_p components with one core and the remaining components with another core.

6.7.2 Scalability and Distributed Performance

Scalability and distributed performance are important for surveillance systems. We quantify those aspects by the impact of adding an additional video workstation or control workstation on the network bandwidth and the surveillance system frame rate. The use of multicast instead of unicast in the proposed architecture for communicating surveillance information and video increases the scalability and distributed performance of the system. We demonstrate this in Fig. 6.13 in terms of network bandwidth needs after adding additional video and control workstations. With multicast, packet duplication is done efficiently at the network router. Increasing the number of control workstations does not increase the bandwidth requirement of the video workstation because the surveillance video is sent only once. This is an important results for systems utilizing many cameras. It also reduces the processing requirements on video workstations. Also, the proposed system allows for adding additional control workstation efficiently, e.g., when using mobile control workstations.

In Fig. 6.14, we examine the effect of adding control workstations on the frame rate

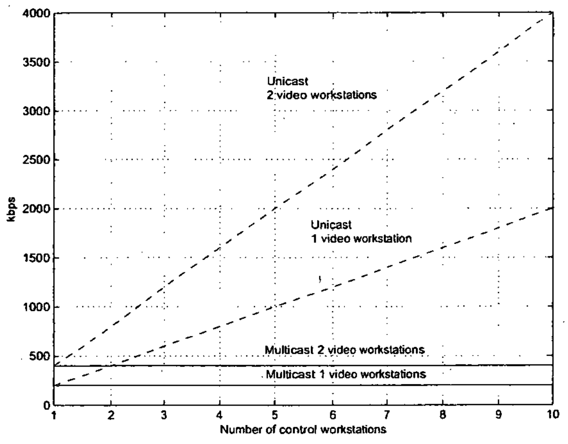


Figure 6.13: Multicast increases the scalability and network efficiency of the surveillance system. Adding additional control workstations does not increase the bandwidth requirements of video workstations.

of the proposed system compared to the system in [9]. In this test, we used the same video processing, streaming, encoding, and archiving used in our system and varied only the architecture. It can be seen from Fig. 6.14 that since the surveillance server of the system in [9] is a central point of the system (instead of an non-central point as in the proposed system) the more control workstation are connected, the lower the frame rate that the server can deliver to them. This is not a major drawback in case of a single control workstation. However, adding mobile control workstations poses a challenge for the system in [9] compared to ours. We tested our proposed system on a WiFi capable mobile device and show the output in Fig. 6.15. The system provides access to all features of the control workstations except for access to multiple surveillance feeds at the same time due to the limited size of the device display and limited CPU power to decode more than one incoming MPEG4 stream.

6.7.3 Testing the Overall System

We use the precision-recall curve to evaluate the performance of the whole system. The closer the curve to 1, the better the performance of both the video object extrac-

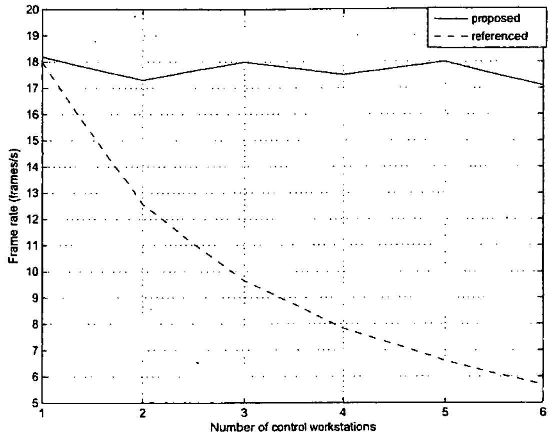


Figure 6.14: The effect of adding control workstations on the frame rate of the proposed system and that of [9].

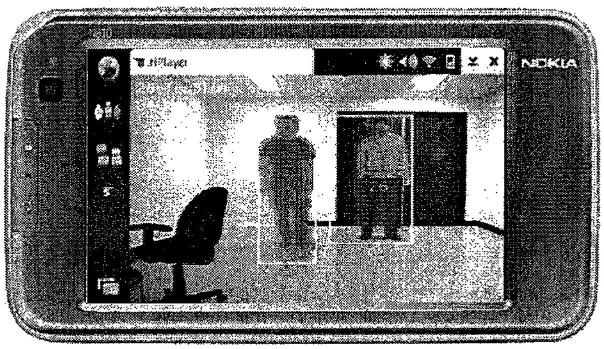
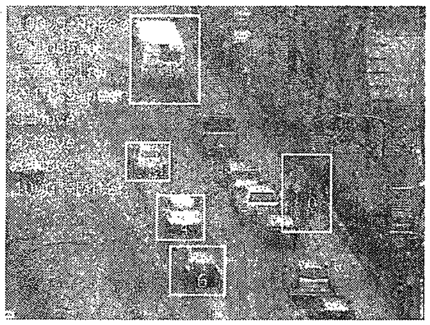


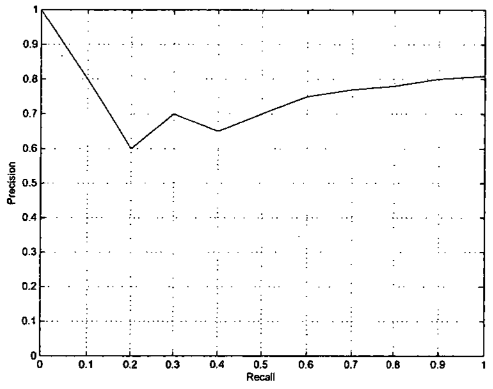
Figure 6.15: Video surveillance on a mobile device.

tion part and the management, archiving, and retrieval of surveillance information part. Precision is defined as the number of relevant results retrieved by a search divided by the total number of results retrieved by that search. Recall is defined as the number of relevant results retrieved by a search divided by the total number of existing relevant results,

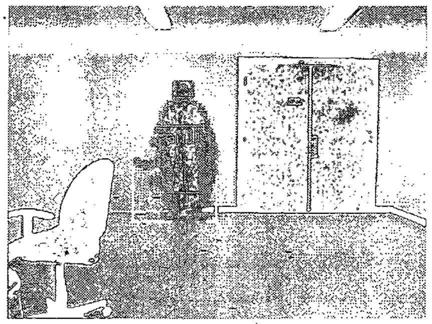
$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \tag{6.5}$$

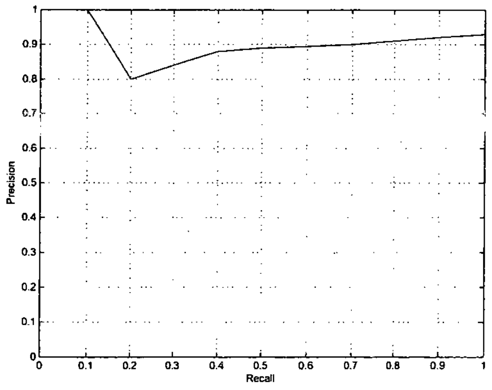
where TP is true positive, and FP is false positive, and FN is false negative. An example object retrieval is as follows: to search for all large objects (e.g., buses) passing through the corner (e.g., exiting a street) of the frame as in Fig. 6.16(a). The resulting precision-recall curve is shown in 6.16(b) with the precision maintained mostly above 70%. An example event retrieval is to search for all events of objects being detected inside a restricted area as in Fig. 6.16(c). The inside-event is an important one which helps localize objects, e.g., in case of vandalism detection, or for counting objects (e.g., counting cars in highway surveillance). Improved precision results (maintained above 80%) are obtained for this query. The retrieval result overall satisfies the requirement of the query [124].


(a) Objects Retrieval



(b) Precision-recall curve

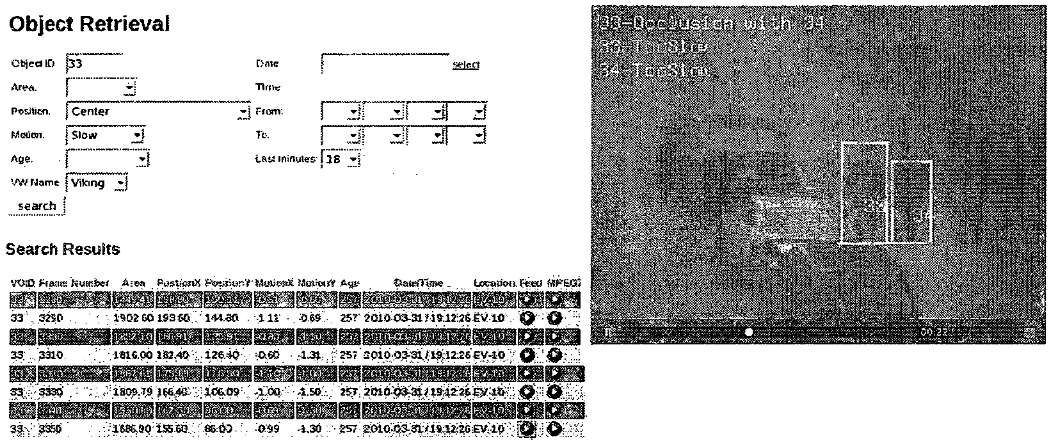

(c) Events Retrieval



(d) Precision-recall curve

Figure 6.16: Retrieval of objects and events. A curve closer to 1 indicates better performance.

In the final experiment, we use our surveillance system to investigate a situation in which a number of repeated object occlusions and expositions is detected in front of a restricted area. This can be viewed as an alarming situation or a possible fight scenario. First, we use the proposed system to find slow moving objects in the center of the scene in the last 18 minutes of the video signal. We locate object ID 33. We then search for occlusion events associated with object ID 33 and observe that as the two pedestrian closely walked up the street, they occluded a number of times. This experiment is shown in Fig. 6.17. We can see that the event retrieval results allow



Object Retrieval

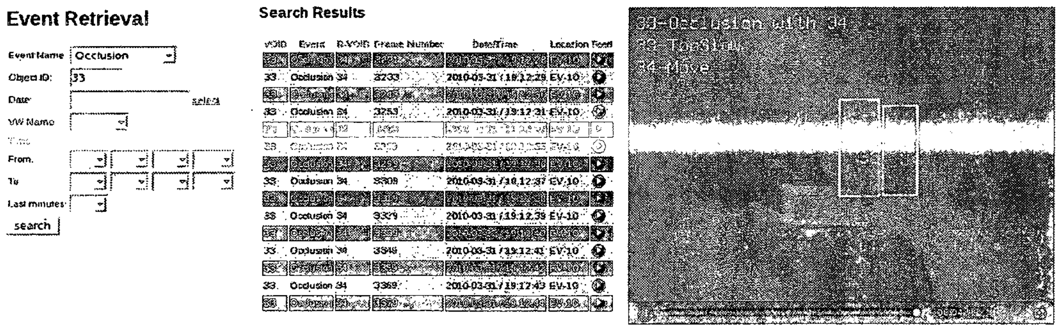
Object ID	33	Date	select
Area.		Time	
Position.	Center	From:	
Motion.	Slow	To.	
Age.		Last minutes	18
VW Name	Viking		

search

Search Results

VOID	Frame Number	Area	PositionX	PositionY	MotionX	MotionY	Age	Date/Time	Location	Feed	MPEG7
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	257	[illegible]	[illegible]		
33	3250	1902.60	193.60	144.80	-1.11	-0.69	257	2010-03-31 / 19:12:26	EV-10		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	EV-10		
33	3310	1816.00	182.40	126.40	-0.60	-1.31	257	2010-03-31 / 19:12:26	EV-10		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	257	[illegible]	[illegible]		
33	3330	1809.79	166.40	106.09	-1.00	-1.50	257	2010-03-31 / 19:12:26	EV-10		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
33	3350	1686.90	155.60	86.00	-0.99	-1.30	257	2010-03-31 / 19:12:26	EV-10		

(a) Objects Retrieval



Event Retrieval

Event Name	Occlusion
Object ID:	33
Date	select
VW Name	
Time	
From.	
To	
Last minutes	

search

Search Results

VOID	Event	R-VOID	Frame Number	Date/Time	Location	Feed
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	Occlusion	34	3233	2010-03-31 / 19:12:29	EV-10	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	Occlusion	34	3253	2010-03-31 / 19:12:31	EV-10	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	[illegible]	[illegible]	[illegible]	[illegible]	EV-10	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	Occlusion	34	3309	2010-03-31 / 19:12:37	EV-10	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	Occlusion	34	3329	2010-03-31 / 19:12:39	EV-10	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	Occlusion	34	3349	2010-03-31 / 19:12:41	EV-10	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
33	Occlusion	34	3369	2010-03-31 / 19:12:43	EV-10	
33	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

(b) Events Retrieval

Figure 6.17: Use of the proposed surveillance system in investigating multiple object occlusions as a sign of fighting in front of a restricted area. Object-based search is used first to locate slow moving objects in the center of the scene. The ID of the video object is then used to search for related occlusion events.

us to decide if the situation is not an alarming one.

6.8 Summary

In this chapter, we proposed a scalable, distributed and real-time surveillance system with for indexing and retrieval capabilities. The system architecture consists of three main modules, namely, video workstations for video processing and indexing, control workstations for viewing and control, and a surveillance server for system management and archiving. The video workstation used a series of segmentation, tracking, and event detection algorithms to index the surveillance video and stream the surveillance video and extracted information to a mutlicast address joined by the server for archiving of surveillance information and video and by control workstation for viewing and control. This increases the scalability and distributed performance of the system. The proposed system reduced storage requirements by modeling object features as temporal Gaussians and used observed variations in object features to find the temporal separation between the Gaussians (to sample features). The proposed system achieves a frame rate of 18 frames/second for SIF video and is real-time due to the proposed indexing scheme (i.e., extraction, delivery and archiving of surveillance information). We used the precision-recall curve to evaluate the indexing and retrieval capabilities of the system. The efficient design of the proposed system enabled us to run it on mobile devices through wireless networks to provide security personnel with access to surveillance video and information both real-time and offline on the move. We compared our architecture with a recently presented one and found that ours is more scalable, network efficient, and achieves higher frame rates.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Video surveillance has attracted the attention of both academia and the industry. Many video surveillance systems either focus on the video processing algorithms to deal with the problems faced towards extracting video objects (i.e., spatio-temporally localizing them and interpreting their behavior while dealing with the challenges brought forth by a temporally varying and content rich signal), or on the system architecture and how to communicate, manage, and index extracted video objects to achieve automated surveillance and retrieval. Few attempts [125–127] are made to investigate both video object extraction and the surveillance system architecture at the same time and study the challenges faced towards building an end-to-end automated surveillance system. Due to the complexity of video surveillance signals, many video processing algorithms are needed, leading to many difficulties in bringing them together. Nevertheless, the end product is not only of industrial value, but of academic one. It provides a framework for testing and improving the various algorithms used and reveals the challenges still ahead.

This thesis has developed a video object extraction engine and integrated it into a distributed, scalable and fast surveillance system architecture. The result of this integration is an end-to-end surveillance system, that can be used to realize many video surveillance applications. One such application demonstrated in this thesis is vandalism detection. The thesis has studied and proposed solutions to the problems associated with video object extraction such as noise estimation, segmentation with fixed and moving cameras, multiple video object tracking with occlusion and split handling, and high-level behavior interpretation. It has also studied and proposed solutions to the problems associated with the communication, indexing, and retrieval of the extracted video objects and the distribution of tasks on the module and component level to realize the potential of the video extraction engine in automated surveillance.

The proposed surveillance system architecture decentralizes the server and stops it from being a single point of failure in the system, and is hence more reliable than conventional architectures. It also shifts the responsibility of broadcasting surveillance video and information from the video processing module to the network, and thus frees the module to focus on processing and achieve higher frame rates. In the proposed architecture, temporal Gaussians are used to model object features and Nyquist sampling theorem is used to decide on the proper separation between the temporal Gaussians. With the proposed model, the storage requirement is significantly reduced. This, however, creates a delay of one third of a second for 30 frames per second video sequences between the object features and the event. This delay is insignificant given the normal speed of objects in surveillance sequences, making the proposed model an improvement over storing extracted objects every frame. Our study has also revealed that to increase the output frame rate of the system, we must aim to have the time needed for processing match that of compression and streaming. We also realized the benefits on the final frame rate coming from running module components on parallel using a multi-core computer. We conclude that higher frame rates can be achieved by researching how to run the video processing algorithms themselves in parallel, which is becoming an emerging trend. We experimented with OpenMP, an application programming interface for parallel programming, by running a processing intensive benchmark test on an 8 core AMD Opteron processor (each core 1000 MHz) and on a 2 core AMD Opteron processor (each core 2192 MHz). The test finished on the 8 core slower processor in approximately half the time it needed to finish on the faster 2 core processor.

From researching noise estimation techniques, we conclude that block-based noise estimation algorithms combine accuracy and fast performance, but rely on large quantities of blocks and homogeneity measurements to achieve this. Reducing the quantity of blocks leads to faster performance at the expense of a loss in accuracy. We proposed using particle filtering techniques to reduce the quantity of blocks without losing accuracy by allowing these blocks to be dynamic as they seek nearby homogeneity. We investigated the application of these homogeneity seeking blocks in noise estimation and they showed promising results. This motivates us to further research their application in segmentation.

Our fast segmentation approach combines the benefits of both background subtraction and background update and yields temporally stable objects for tracking. The price of this stability is increased sensitivity to shadows, which translates to false occlusions. These false occlusions are partial in nature and do not pose a challenge for the proposed object tracking, apart from increased computations. On the other hand, temporally unstable objects lead to track loss, which is a more serious problem. Moreover, the proposed approach is not as robust at night as it is during the day because it is challenged, as most segmentation techniques are, by low illumination and low contrast sequences. How to pre-process the surveillance signal to improve night segmentation is an interesting research topic. To handle moving cameras, we used local frame neighborhoods to generate background models and use them to segment objects. This allowed us to achieve accurate segmentation results, but is computationally expensive for real-time object tracking. We reduced the computational complexity of the algorithm. Despite this, there exist object trackers that require no segmentation. These algorithms rely on object detection to initialize tracking, but assume no global motion. The proposed approach helps in cases where the tracking needs to be initialized during global motion. There are also other application areas such as coding where object segmentation in global motion is very crucial.

Multiple video object tracking is a very challenging task, especially in a crowded scene with different object scales. Both deterministic (e.g., rule-based) and probabilistic trackers have their advantages and disadvantages. The main advantage the deterministic trackers have over probabilistic trackers is the fact that increasing the number of objects is far less computationally inexpensive than with probabilistic tracker due to the increase in dimensionality. This is important for building the proposed surveillance system. The main advantage of probabilistic trackers is the fact that they do not require segmentation and can handle moving cameras easily. Moreover, probabilistic trackers need prior knowledge of the models of objects to track in the initialization stage. This is often done manually, which is not practical. Alternatively, a deterministic (i.e., a detection or rule-based) method, like the one proposed, is needed to identify and track moving objects until they stabilize and then initialize the probabilistic tracking stage. Therefore, a better approach to tracking is to perhaps integrate both types of trackers.

Methods for vandalism detection use either the detection of abnormal behavior of vandals or the detection of changes caused by vandalism. The proposed vandalism detection method uses both. Since the detection of abnormal or strange behavior may not be accurate or reliable because of the difficulty in trying to define what constitutes a strange pattern in the deviations of object features, we base our algorithm on basic events (e.g., inside, moving, staying for long, etc). At the same time, we also monitor the changes caused by vandalism. This is important because suppose, for example, a sick person is moving close to a restricted site. This may be detected as abnormal movements or variation of features, however, it is not enough to detect vandalism without considering the changes to the site.

7.2 Future Work

There are several extensions to our proposed work. In segmentation, a possible extension is to integrate shadow detection and removal into the segmentation process. While the fast video object tracking used took advantage of the temporally stable and clean blobs produced by the fast segmentation approach, this came at the expense of increased sensitivity to shadows. There are algorithms which can detect and remove shadows reducing false occlusions and improving the overall tracking reliability. Moreover, there have been recent advances in the area of night segmentation by preprocessing the night time signal before segmentation. Using a fast method to night segmentation is interesting as it allows automated outdoors surveillance to continue at night, when it is most needed.

In global motion estimation, a possible extension is to investigate if the output produced from the proposed global motion segmentation technique can improve the outliers detection process. Another extension is to investigate how to prevent noise from interfering with global motion estimation and how to reduce the effect of error propagation when predicting long-term motion parameters from short-term ones.

In tracking, a possible extension is to produce a hybrid tracking method which switches from multiple video object tracking to particle filtering based single video object tracking during global motion, then switches back to the proposed multiple video object tracking. This allows the tracker to keep objects already tracked during global motion, but does not initiate tracking for newly appearing objects. The proposed global motion segmentation method can help detect and initiate tracking for those newly appearing objects, which motivates us to further investigate reducing its complexity.

In vandalism detection, a possible extension is the use of face detection and recognition of vandals to prevent repeated vandalism cases. Also, post-vandalism vandal tracking can be key in providing authorities with crucial information (e.g., the vandal's car make and model as they flee the scene). This requires switching from multiple to single object tracking and coordinating camera controls with the output of tracking. Since vandalism often occurs during the night and the proposed approach for vandalism detection relies on segmentation, a robust night segmentation approach will enable vandalism detection to work at night.

7.3 Research Papers

Submitted:

- M. Ghazal and A. Amer, "Scalable Distributed Video Surveillance System with Object Feature Variation Modeling using Temporal Gaussians," *Springer Journal on Signal Process. Systems*, 2009.

- M. Ghazal, C. Vázquez and A. Amer, "Real-time Vandalism Detection By Monitoring Object Activities," *Springer Journal on Multimedia Tools and Applications*, 2009.

- M. Ghazal and A. Amer, "Noise Estimation in Images and Video Frames using Particle Filtering For Spatial Homogeneity Localization," *IEEE Trans. on Image Process.*, 2009 (Accepted).

Published:

- C. Vázquez, M. Ghazal, and A. Amer, "Feature-based detection and correction of occlusions and split of video objects," *Springer Journal on Signal, Image and Video Process.*, vol. 3, no. 1, pp. 13-25, 2009.
- M. Ghazal and A. Amer, "Total occlusion correction using invariant wavelet features," in *Proc. IEEE Int. Conf. Image Process.*, pp. 345-348, 2007.
- M. Ghazal, C. Vazquez, and A. Amer, "Real-time automatic detection of vandalism behavior in video sequences," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, pp. 1056-1060, 2007.
- D. Ostheimer, S. Lemay, D. Mayisela, P. Dagba, M. Ghazal, and A. Amer, "A Modular Distributed Surveillance System Over IP," in *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering*, pp. 1001-1004, 2006.

References

[1] D.-H. Shin, R.-H. Park, Y. Seungjoon, and J.-H. Jung, "Block-based noise estimation using adaptive Gaussian filtering," *IEEE Trans. on Consumer Electronics*, vol. 51, no. 1, pp. 218–226, February 2005.

[2] A. Amer and E. Dubois, "Fast and reliable structure-oriented video noise estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 113–118, January 2005.

[3] J. Yang, Y. Wang, W. Xu, and Q. Dai, "Image and video denoising using adaptive dual-tree discrete wavelet packets," *Circuits and Systems for Video Technology, IEEE Trans. on*, vol. 19, no. 5, pp. 642 –655, may 2009.

[4] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch-based image sequence restoration," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 29, no. 6, pp. 1096 –1102, june 2007.

[5] F. Achkar and A. Amer, "Hysteresis-based selective gaussian mixture models for real-time background maintenance," in *Proc. SPIE Visual Communications and Image Process.*, 2007, vol. 6508, pp. J1–J11.

[6] A. Krutz, A. Glantz, T. Borgmann, M. Frater, and T. Sikora, "Motion-based object segmentation using local background sprites," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing*, 2009, pp. 1221–1224.

[7] A. Amer, "Voting-based simultaneous tracking of multiple video objects," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 1448–1462, Nov. 2005.

[8] V. Mezaris, I. Kompatsiaris, and M. G. Strintzis, "Video object segmentation using bayes-based temporal tracking and trajectory-based region merging," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 782–795, June 2004.

[9] S. Hosik, E. T. AnzaKu, W. De Neve, Yong Man Ro, and K. N. Plataniotis, "Privacy protection in video surveillance systems using scalable video coding," in *Proc. IEEE Int. Conf. on Advanced Video and Singal Based Surveillance*, 2009, pp. 424–429.

[10] T. Yang, S. Z. Li, Q. Pan, and J. Li, "Real-time multiple objects tracking with occlusion handling in dynamic scenes," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2005, vol. 1, pp. 20–25.

[11] J. Sauvé, "Crime statistics in canada," *Statistics Canada - Catalogue no. 85-002-XPE*, vol. 25, no. 5, 2004.

[12] M. Barker and C. Bridgeman, "Preventing vandalism: what works?!," *Police Research Group, Crime Detection and Prevention Series: Paper No. 56*, pp. 1–3, 1994.

[13] N. Ning and T. Tan, "A framework for tracking moving target in a heterogeneous camera suite," in *Proc. Int. Conf. Control, Automation, Robotics and Vision*, 2006, pp. 1–5.

[14] C. Vazquez, M. Ghazal, and A. Amer, "Feature-based detection and correction of occlusions and split of video objects," *Springer Journal on Signal, Image and Video Process.*, vol. 3, no. 1, pp. 13–25, 2009.

[15] M. Ghazal and A. Amer, "Homogeneity localization using particle filters with application to noise estimation," *IEEE Trans. on Image Process.*, 2010, (ACCEPTED).

[16] M. Ghazal, C. Vazquez, and A. Amer, "Real-time automatic detection of vandalism behavior in video sequences," in *IEEE Int. Conf. Syst. Man Cybern.*, 2007, pp. 1056–1060.

[17] D. Ostheimer, S. Lemay, D. Mayisela, P. Dagba, M. Ghazal, and A. Amer, "A modular distributed surveillance system over ip," in *IEEE Canadian Conf. on Electrical and Computer Engineering*, 2006, pp. 1001–1004.

[18] M. Ghazal and A. Amer, "Total occlusion correction using invariant wavelet features," in *IEEE Int. Conf. Image Process.*, 2007, pp. 345–348.

[19] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, "Noise estimation from a single image," in *Proc. IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition*, 2006, pp. 901–908.

[20] A. Bovik, *Handbook of Image and Video Processing*, pp. 275–285, Elsevier Academic Press, second edition, 2005.

[21] B. C. Song and K. W. Chun, "Noise power estimation for effective de-noising in a video encoder," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 357–360, March 2005.

[22] B. C. Song and K. W. Chun, "Motion-compensated noise estimation for efficient pre-filtering in a video encoder," *IEEE Int. Conf. on Image Processing*, vol. 2, pp. 211–214, September 2003.

[23] S. I. Olsen, "Estimation of noise in images: an evaluation," *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 4, pp. 319–323, July 1993.

[24] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, April 1994.

[25] E. J. Balster, Y. Zheng, and R. Ewing, "Combined spatial and temporal domain wavelet shrinkage algorithm for video denoising," *IEEE Trans. on Circuits and Systems for Video Technology*. vol. 16, no. 2, pp. 220–230, February 2006.

[26] J. Boulanger, J.-B. Sibarita, C. Kervrann, and P. Bouthemy, "Non-parametric regression for patch-based fluorescence microscopy image sequence denoising," in *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE Int. Sym. on*, 14-17 2008, pp. 748 –751.

[27] G. de Haan, T. G. Kwaaitaal-Spassova, M. Larragy, and O. A. Ojo, "Memory integrated noise reduction IC for television," *IEEE Trans. on Consumer Electronics*, vol. 42, no. 2, pp. 175–181, 1996.

[28] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 146, no. 2, pp. 80–84, April 1999.

[29] R. C. Bilcu and M. Vehvilainen, "A new method for noise estimation in images," *IEEE EURASIP International Workshop on Nonlinear Signal and Image Processing*, p. 25, May 2005.

[30] J. S. Lee and K. Hoppel, "Noise modeling and estimation of remotely-sensed images," *Proceedings of the* 12^{th} *Int. Geoscience and Remote Sensing Sym.*, vol. 2, pp. 1005–1008, July 1989.

[31] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal on Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[32] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–189, 2002.

[33] Z. Wang, X. Yang, Y. Xu, and S. Yu, "Camshift guided particle filter for visual tracking," *Pattern Recognition Letters*, vol. 30, no. 4, pp. 407–413, 2009.

[34] C. Shan, T. Tan, and Y. Wei, "Real-time hand tracking using a mean shift embedded particle filter," *Pattern Recognition*, vol. 40, no. 7, pp. 1958–1970, 2009.

[35] R. Douc and O. Cappe, ," in *Comparison of resampling schemes for particle filtering*, 2005, pp. 64–69.

[36] J. Immerkær, "Fast noise variance estimation," *Computer Vision and Image Understanding*, vol. 64, no. 2, pp. 300–302, September 1996.

[37] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, pp. 127–129, Prentice Hall, 2002.

[38] E. Durucan and T. Ebrahimi, "Moving object detection between multiple and color images," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, July 2003, pp. 243–251.

[39] L. D. Stefano, S. Mattoccia, and M. Mola, "A change-detection algorithm based on structure and color," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, July 2003, pp. 252–259.

[40] Y. Hwang, J. S. Kim, and I. Kweon, "Change detection using a statistical model of the noise in color images," in *Proc. IEEE Int. Conf. Intell. Robots and Systems*, Oct. 2004, vol. 3, pp. 2713–2718.

[41] T. Alexandropoulos, S. Boutas, V. Loumos, and E. Kayafas, "Real-time change detection for surveillance in public transportation," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Sept. 2005, pp. 58–63.

[42] C. Su and A. Amer, "A color based gray-level compensation algorithm for fast change detection," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing*, 2007, pp. I–1089–1092.

[43] D. Tsai and S. Lai, "Independent component analysis-based background subtraction for indoor surveillance," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 158–167, 2009.

[44] P. L. Rosin, "Thresholding for change detection," in *Proc. IEEE Int. Conf. Computer Vision*, 2002, vol. 86, pp. 79–95.

[45] P. L. Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection," *Pattern Recognition Letters*, vol. 24, pp. 2345–2356, 2002.

[46] P. L. Rosin and T. Ellis, "Image difference threshold strategies and shadow detection," in *Proc. Britich Machine Vision Conf.*, 1995, pp. 347–356.

[47] J. Kapur, P. Sahoo, and A. Wong, "A new method for gray-level picture thresholding using the entropy of histogram," *Computer Vision, Graphics, and Image Process.*, vol. 29(3), pp. 273–285, 1985.

[48] A. Pikaz and A. Averbuch, "Digital image thresholding based on topological stable state," *Pattern Recognition*, vol. 29, pp. 829–843, 1996.

[49] C. Su and A. Amer, "A real-time adaptive thresholding for video change detection," in *Proc. IEEE Int. Conf. Image Processing*, 2006, pp. 157–160.

[50] L. Jia and Y. Liu, "A novel thresholding approach to background subtraction," in *Proc. IEEE. Workshop on Applications of Computer Vision*, 2008, pp. 1–6.

[51] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *Proc. IEEE Int. Conf. Pattern Recognition*, 2008, pp. 1–4.

[52] Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from video," in *Proc. Int. Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.

[53] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 780–785, 1997.

[54] C. Stauffer and W. EE. I. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 1999, vol. 2, p. 252.

[55] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proc. European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[56] Z. Zivkovic, "An improved adaptive gaussian mixture model for background subtraction," in *Proc. Int. Conf. Pattern Recognition*, 2004, pp. 28–31.

[57] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 5, pp. 827–832, 2005.

[58] Z. Wei, S. Jiang, and Q. Huang, "A pixel-wise local information-based background subtraction approach," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2008, pp. 1501–1504.

[59] J. M. McHugh, J. Konrad, V. Saligrama, and P.-M. Jodoin, "Foreground-adaptive background subtraction," *Proc. IEEE*, vol. 16, no. 5, pp. 390–393, 2009.

[60] Z. Li, P. Jiang, H. Ma, J. Yang, and D. Tang, "A model for dynamic object segmentation with kernel density estimation based on gradient features," *Image Vision Comput.*, vol. 27, no. 6, pp. 817–823, 2009.

[61] R. Mech and M. Wollborn, "A noise robust method for 2d shape estimation of moving objects in video sequences considering a moving camera," *Signal Processing (special issue)*, vol. 66, no. 2, pp. 213–217, 1998.

[62] F. Dufaux and K. Konrad, "Efficient, robust and fast global motion estimation for video coding," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 497–501, 2000.

[63] B. Qi, M. Ghazal, and A. Amer, "Robust global motion estimation oriented to video object segmentation," *IEEE Trans. Image Process.*, , no. 6, pp. 958–967, June 2008.

[64] B. Qi, "Fast and robust global motion estimation in video object segmentation," *M. A. Sc. thesis, Concordia University*, pp. 44–67, March 2005.

[65] D. Farin and P. de With, "Enabling arbitrary rotational camera motion using multi-sprites with minimum coding cost," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 492–506, 2006.

[66] M. Unger, M. Asbach, and P. Hosten, "Enhanced background subtraction using global motion compensation and mosaicing," in *Proc. IEEE Int. Conf. Image Processing*, 2008, pp. 2708–2711.

[67] S. Chien, Y. Huang, B. Hsieh, S. Ma, and L. Chen, "Fast video segmentation algorithm with shadow cancellation, global motion compensation, and adaptive threshold techniques," *IEEE Trans. Multimedia*, vol. 6, no. 5, pp. 732–748, 2006.

[68] A. Amer, "Memory-based spatio-temporal real-time object segmentation," in *Proc. SPIE Real-time Imaging*, 2003, vol. 5012, pp. 10–21.

[69] C. Su, "Fast and artifact-robust change detection with video-content assessment," *Masters thesis, Concordia University*, pp. 21–25, November 2006.

[70] A. Amer and E. Dubois, "Image segmentation by robust binarization and fast morphological edge detection," in *in Proc. IAPR/CIPPRS Int. Conf. on Vision Interface*, 2000, pp. 357–364.

[71] D. J. Bullock and J. S. Zelek, "Real-time tracking for visual interface applications in cluttered and occluding situations," *Image and Vision Computing*, vol. 22, pp. 1083–1097, Oct. 2004.

[72] H. T. Nguyen and A. W. M. Smeulders, "Fast occluded object tracking by robust appearance filter," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, pp. 1099–1104, Aug. 2004.

[73] L. Ruijiang, L. Liyuan, H. Weimin, and S. Qibin, "Multi-strategy object tracking in complex situation for video surveillance," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 2008, pp. 2749–2752.

[74] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Tracking video objects in cluttered background," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, pp. 575–584, Apr. 2005.

[75] S. Piva, L. Marchesotti, and C. S. Regazzoni, "A dynamic model integrating colour and shape information for object tracking in conditions of occlusion," in *Proc. IEEE Int. Conf. Multimedia and Expo*, June 2003, vol. 3, pp. 1547–1550.

[76] N. Peterfreund, "Robust tracking of position and velocity with Kalman snakes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 6, pp. 564–569, June 1998.

[77] L. Young-Sook and L. HoonJae, "Multiple object tracking for fall detection in real-time surveillance system," in *Proc. Int. Conf. Advanced Communication Technology*, 2009, pp. 2308–2312.

[78] Ying Wu and Ting Yu, "A field model for human detection and tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, no. 5, pp. 753–765, May 2006.

[79] P. Pan and D. Schonfeld, "Adaptive resource allocation in particle filtering for articulated object tracking," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing*, 2008, pp. 729–732.

[80] P. P. and D. Schonfeld, "Optimal particle allocation in particle filtering for multiple object tracking," in *Proc. IEEE Int. Conf. Image Processing*, 2007, pp. I61–I64.

[81] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *Int. Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[82] J. Rittscher, N. Krahnstoever, and L. Galup, "Multi-target tracking using hybrid particle filtering," in *Proc. IEEE Workshops on Application of Computer Vision*, 2005, vol. 1, pp. 447–454.

[83] K. Nummiaro, E. B. Koller-Meier, and L. Van Gool, "Object tracking with an adaptive color-based particle filter," in *Symp. for Pattern Recognition of the DAGM*, 2002, pp. 355–360.

[84] L. Mihaylova, P. Brasnett, N. Canagarajah, and D. Bull, *Object tracking by particle filtering techniques in video sequences*, IOS press, 2006.

[85] Y. Zhou and H. Tao, "A background layer model for object tracking through occlusion," in *Proc. IEEE Int. Conf. Computer Vision*, Oct. 2003, vol. 2, pp. 1079–1085.

[86] C. M. Pun and M. C. Lee, "Log-polar wavelet energy signatures for rotation and scale invariant texture classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, pp. 590–603, May 2003.

[87] Chao He, Y.F. Zheng, and S.C. Ahalt, "Object tracking using the gabor wavelet transform and the golden section algorithm," *Multimedia, IEEE Transactions on*, vol. 4, no. 4, pp. 528–538, dec. 2002.

[88] C. Erdem, B. Sankur, and A. Tekalp, "Performance measures for video object segmentation and tracking," *IEEE Trans. Image Process.*, vol. 13, no. 7, pp. 937–951, July 2004.

[89] J. Popoola and A. Amer, "Performance evaluation for tracking algorithms using object labels," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing*, 2008, pp. 733–736.

[90] D. Taoa, Y. Yuanb, J. Shenc, K. Huangd, and X. Lie, "Visual information analysis for security," *Signal Processing*, vol. 89, no. 12, pp. 2311–2312, 2009.

[91] M. Valera and S. A. Velastin, "Intelligent distributed surveillance systems: a review," *IEE Vision, Image and Signal Process.*, vol. 152, no. 2, pp. 192–204, 2005.

[92] J. Ma, W. Liu, P. Miller, and W. Yan, "Event composition with imperfect information for bus surveillance," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, Sept. 2009, pp. 382 –387.

[93] T. S. K. Chan and K. S. M. Chung, "Applications and selections of intelligent surveillance system in railway industry," in *International Conference on Challenges for Railway Transportation in Information Age*, March 2008, pp. 1 –6.

[94] J. Candamo, M. Shreve, D.B. Goldgof, D.B. Sapper, and R. Kasturi, "Understanding transit scenes: A survey on human behavior-recognition algorithms," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 1, pp. 206 –224, march 2010.

[95] P. Allen, S. Feiner, L. Meskell, K. Ross, A. Troccoli, B. Smith, H. Benko, E. Ishak, , and J. Conlon, "Digitally modeling, visualizing and preserving archaeological sites," in *Proc. Joint ACM/IEEE Conf. on Digital Libraries*, 2004, p. 389.

[96] C. Sacchi, C. Regazzoni, G. Gera, and G. Foresti, "Use of neural networks for behavior understanding in railway transport monitoring applications," in *Proc. Int. Conf. Image Process.*, 2001, pp. 541–544.

[97] C. Sacchi, C. Regazzoni, and G. Vernazza, "A neural network-based image processing system for detection of vandal acts in unmanned railway environments," in *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, Sep 2001, pp. 529 –534.

[98] D. Angiati, G. Gera, S. Piva, and C. Regazzoni, "A novel method for graffiti detection using change detection algorithm," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, 2005, pp. 242–246.

[99] B.C. Chee, M. Lazarescu, and T. Tan, "Detection and monitoring of passengers on a bus by video surveillance," in *Proc. Int. Conf. Image Analysis and Process.*, 2007, pp. 143–148.

[100] F. Tombari, L. Di Stefano, S. Mattoccia, and A. Zanetti, *Graffiti Detection Using a Time-Of-Flight Camera*, pp. 645–654, Springer Berlin / Heidelberg, 2008.

[101] F. Cupillard, A. Avanzi, F. Bremond, and M. Thonnat, "Video understanding for metro surveillance," in *Proc. IEEE Int. Conf. Networking, Sensing and Control*, 2004, pp. 186–191.

[102] L. di Stefano, F. Tombari, A. Lanza, S. Mattoccia, and S. Monti, "Graffiti detection using two views," in *The Eighth Int'l Workshop on Visual Surveillance*, 2008, vol. 1.

[103] H. Weiming, T. Tieniu, W. Liang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. Syst. Man Cybern.*, vol. 34, no. 3, pp. 334–352, 2004.

[104] M. H. Sedky, M. Moniri, and C. C. Chibelushi, "Classification of smart video surveillance systems for commercial applications," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, 2005, pp. 638–643.

[105] A. Amer and C. Regazzoni, "Introduction to the special issue on video object processing for surveillance applications," *Elsevier Journal for Real-Time Imaging*, vol. 11, pp. 1–5, 2005.

[106] M. Magno, F. Tombari, D. Brunelli, L. Di Stefano, and L. Benini, "Multimodal abandoned/removed object detection for low power video surveillance systems," in *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, 2009, pp. 188 –193.

[107] M. Brandel, "A buyer's guide to network cameras," *CSO: Security and Risk*, p. 8, March 2008.

[108] G. Wei, D. Zhang, S. Wu, and Y. Cao, "Design and implementation of an IP-based intelligent video surveillance system," in *Proc. Int. Conf. on Signal Process.*, 2006, vol. 2.

[109] J. R. Black, "Applications of video surveillance technology in airports," in *Proc. Int. Carnahan Conf. on Security Technology*, 2004, pp. 42–46.

[110] F. Ziliani, "The importance of 'scalability' in video surveillance architectures," in *Proc. IEE Int. Symp. on Imaging for Crime Detection and Prevention*, 2005, pp. 29–32.

[111] C. Sacchi and C. S. Regazzoni, "A distributed surveillance system for detection of abandoned objects in unmanned railway environments," *IEEE Trans. Vehicular Technology*, vol. 49, no. 5, pp. 2013–2026, 2000.

[112] S. Kamijo, H. Koo, X. Liu, K. Fujihira, and M. Sakauchi, "Development and evaluation of real-time video surveillance system on highway based on semantic hierarchy and decision surface," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2005, vol. 1, pp. 840–846.

[113] A. Yoneyama, C. H. Yeh, and C.-C.J. Kuo, "Robust traffic event extraction via content understanding for highway surveillance system," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2004, vol. 3, pp. 1679–1682.

[114] H. Shao, L. Li, P. Xiao, and M. Leung, "ELEVIEW: An active elevator video surveillance system," in *Proc. IEEE Workshop on Human Motion*, 2000, pp. 67–72.

[115] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "Computer vision system for in-house video surveillance," *IEE Vision, Image and Signal Process.*, vol. 152, no. 2, pp. 242–249, 2005.

[116] S. Muller-Schneiders, T. Jager, H. S. Loos, and W. Niem, "Performance evaluation of a real time video surveillance system," in *Proc. Int. Conf. Computer Commun. and Networks*, 2005, pp. 137–144.

[117] W. P. Berriss, W. G. Price, and M. Z. Bober, "Real-time visual analysis and search algorithms for intelligent video surveillance," in *Proc. Int. Conf. on Visual Information Engineering*, 2003, vol. 495, pp. 226–229.

[118] X. Yuan, Z. Sun, Y. Varol, and G. Bebis, "A distributed visual surveillance system," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, 2003, pp. 199–204.

[119] H. Dias, J. Rocha, P. Silva, C. Leao, and L.P. Reis, "Distributed surveillance system," in *Proc. Portuguese Conf. on Artificial Intell*, 2005, pp. 257–261.

[120] C. Jaynes, S. Webb, R. Matt Steele, and Q. Xiong, "An open development environment for evaluation of video surveillance systems," in *Proc. IEEE Workshop on Performance Analysis of Video Surveillance and Tracking*, 2002, pp. 32–39.

[121] L. Yan, L. Ren-Fa, X. Cheng, and Y. Fei, "Design and implementation on knowledge discovery and data mining," in *Proc. Int. Workshop Control, Automation, Robotics and Vision*, 2008, pp. 570–573.

[122] A. Amer, E. Dubois, and A. Mitiche, "Rule-based real-time detection of context-independent events in video shots," *Elsevier Journal for Real-Time Imaging*, vol. 11, no. 3, pp. 244–256, 2005.

[123] M.H.-Y. Liao, D.-Y. Chen, C.-W. Sua, and H.-R. Tyan, "Real-time event detection and its application to surveillance systems," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006, pp. 4 pp. –512.

[124] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank, "Semantic-based surveillance video retrieval," *Image Processing, IEEE Transactions on*, vol. 16, no. 4, pp. 1168 –1181, april 2007.

[125] H. Dias, J. Rocha, P. Silva, C. Leao, and L.P. Reis, "Distributed surveillance system," in *Artificial intelligence, 2005. epia 2005. Portuguese conference on*, 5-8 2005, pp. 257 –261.

[126] S. Muller-Schneiders, T. Jager, H.S. Loos, and W. Niem, "Performance evaluation of a real time video surveillance system," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, 15-16 2005, pp. 137 – 143.

[127] M. Saini, M. Kankanhalli, and R. Jain, "A flexible surveillance system architecture," in *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*, 2-4 2009, pp. 571 –576.

Appendix A

Visual Segmentation Comparisons

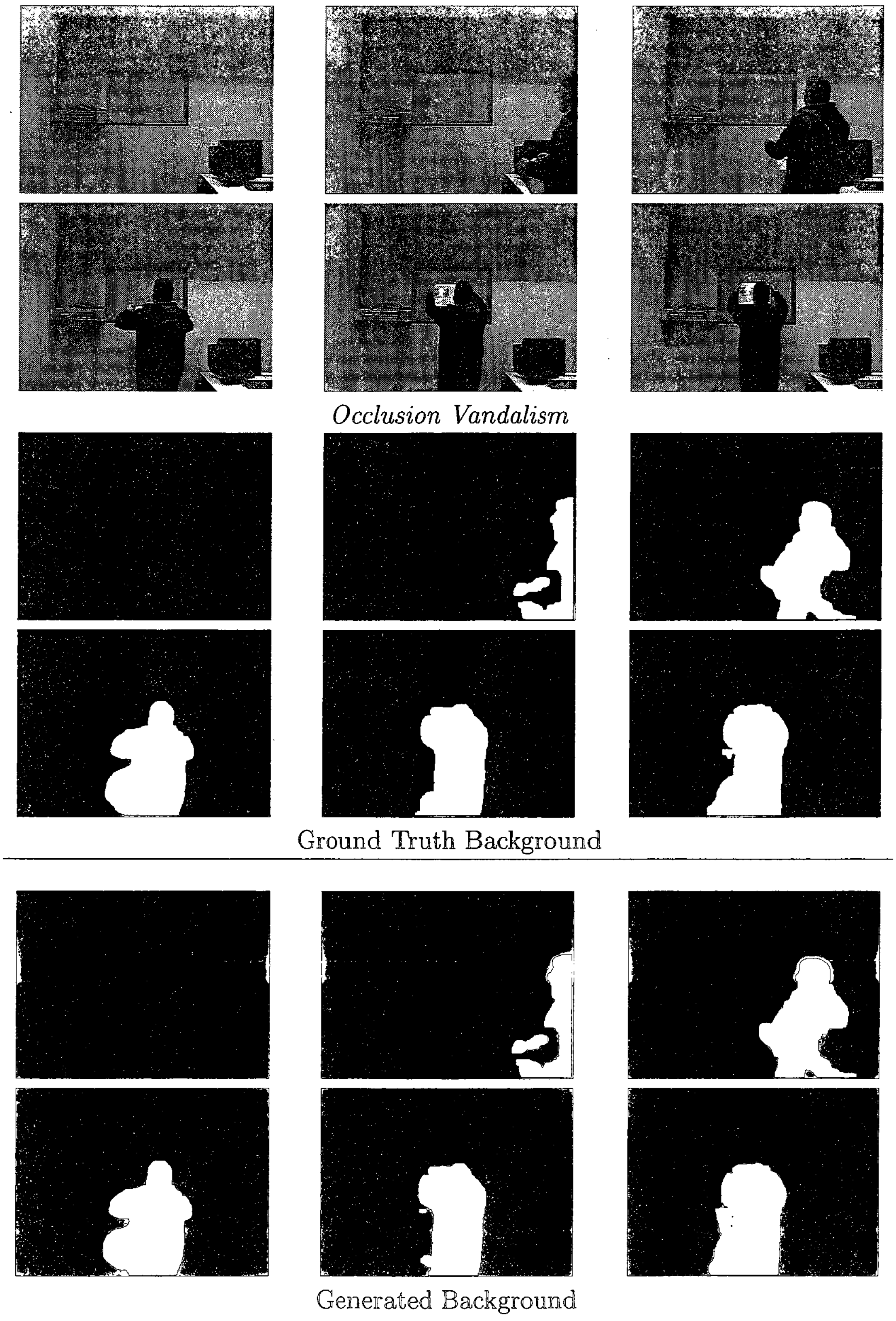
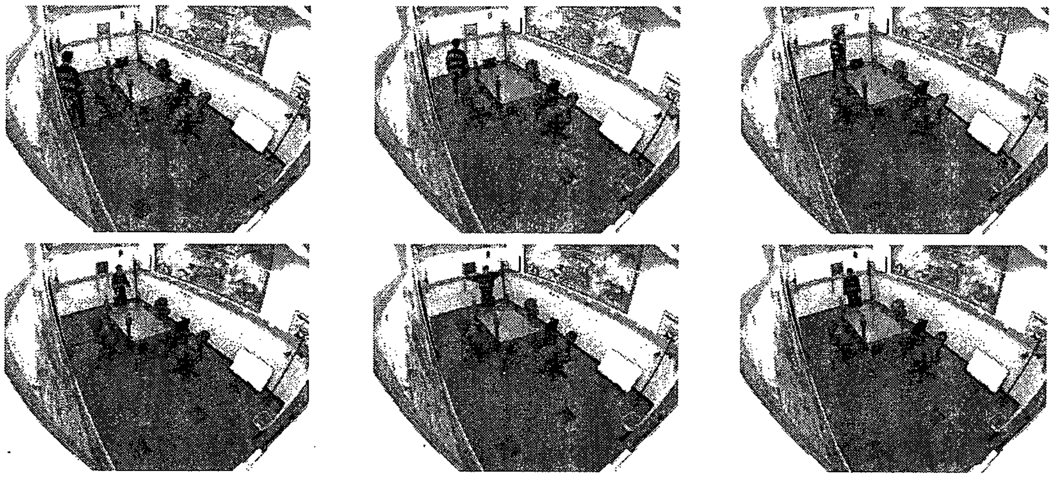
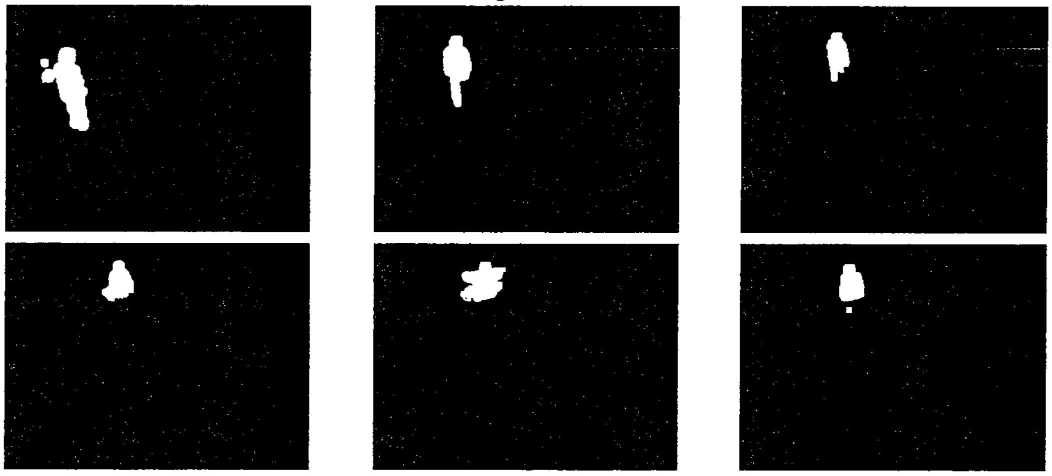


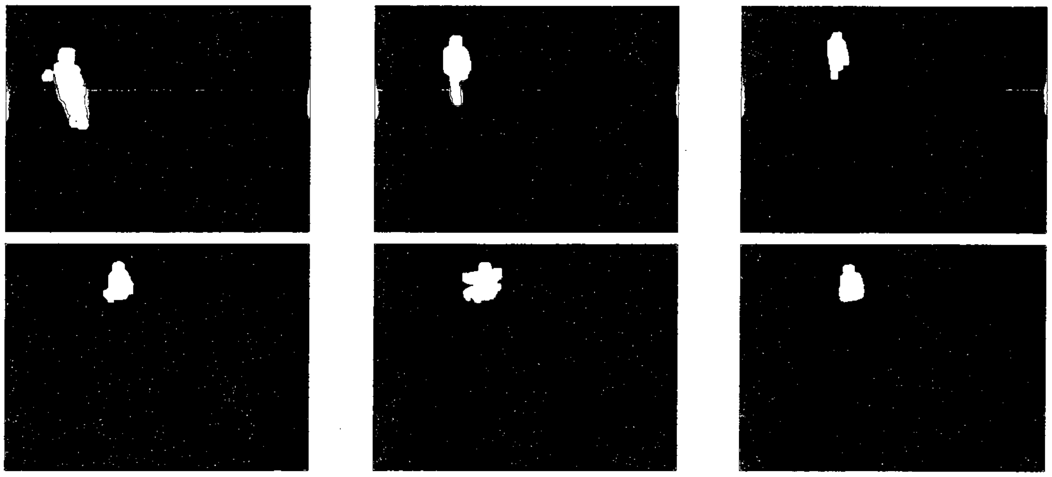
Figure A.1: Subjective comparison of the fast segmentation method with ground truth and automatically (using [5]) generated background models for frames $l = 650$ to 750 of the *Occlusion Vandalism* sequence.



Intelligent Room

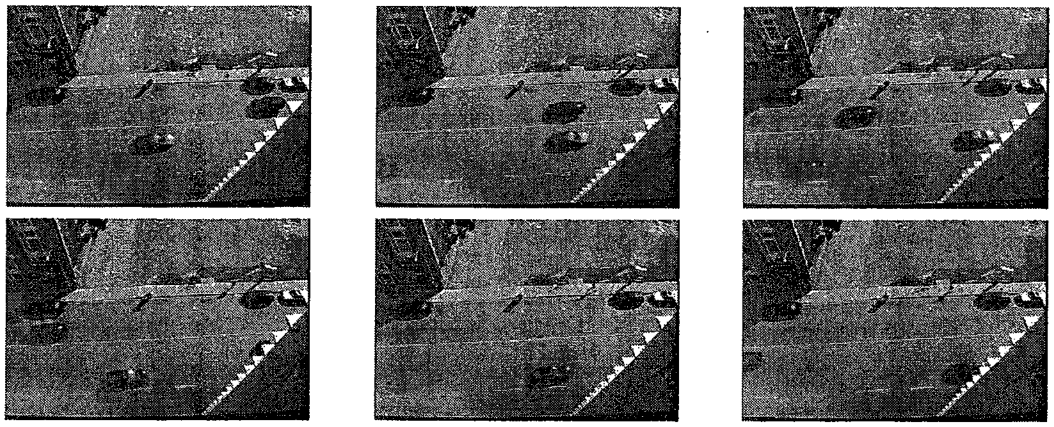


Ground Truth Background

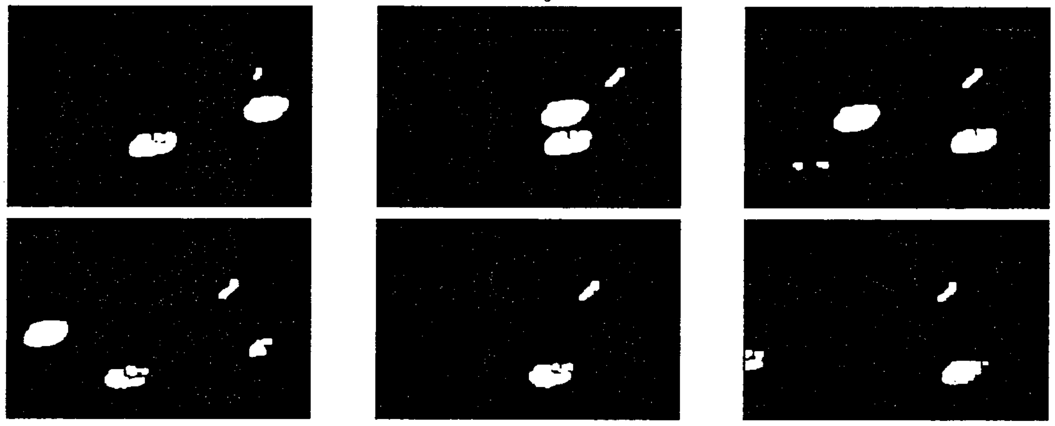


Generated Background

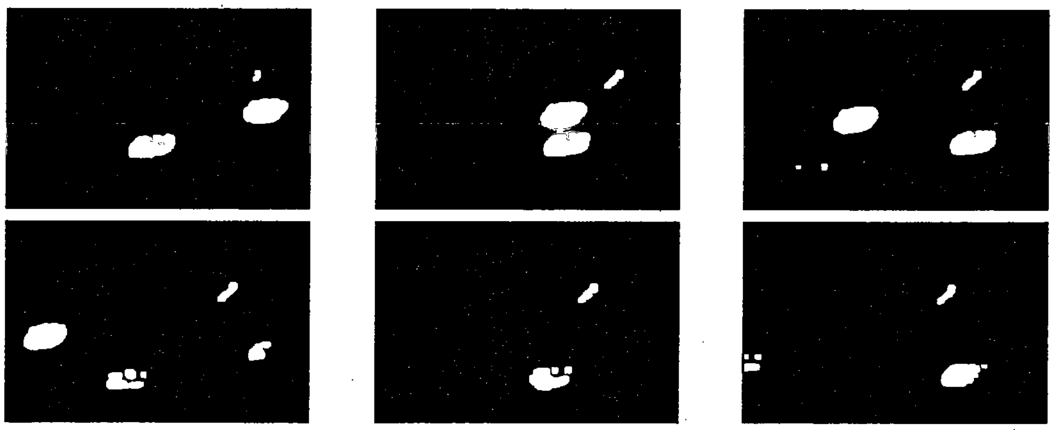
Figure A.2: Subjective comparison of the fast segmentation method with ground truth and automatically (using [5]) generated background models for frames $l = 90$ to 190 of the *Intelligent Room* sequence.



Guy

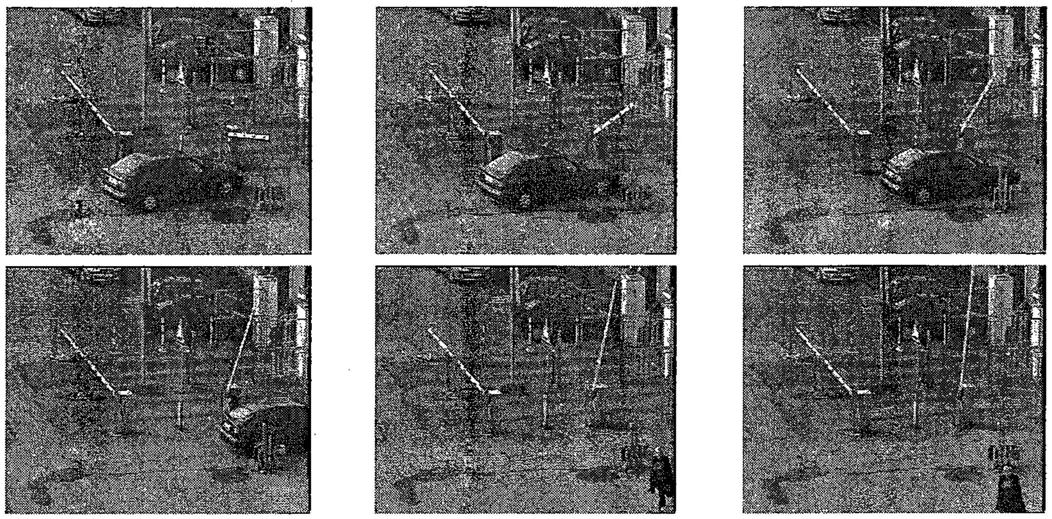


Ground Truth Background

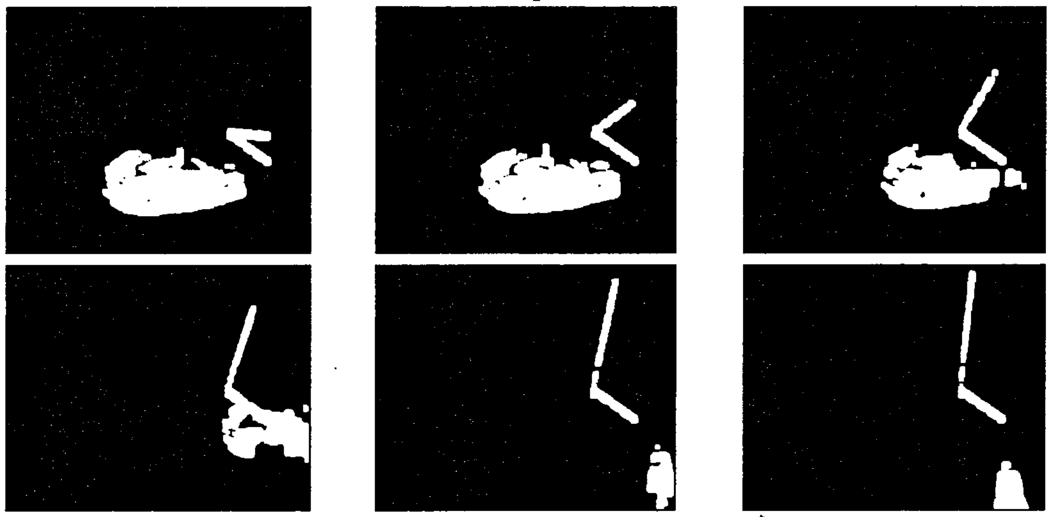


Generated Background

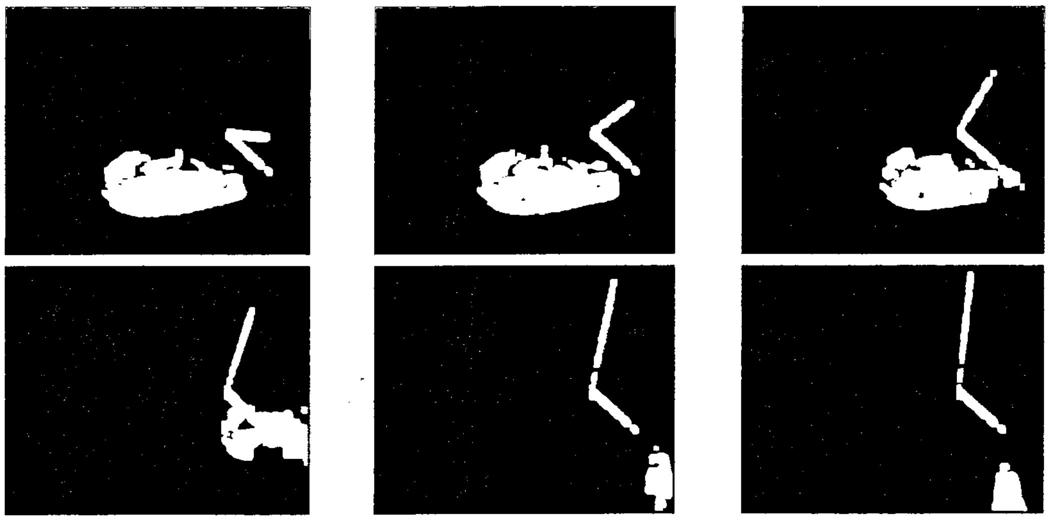
Figure A.3: Subjective comparison of the fast segmentation method with ground and automatically (using [5]) generated background models for frames l = 500 to 600 of the *Guy* sequence.



Campus



Ground Truth Background



Generated Background

Figure A.4: Subjective comparison of the fast segmentation method with ground truth and automatically (using [5]) generated background models for frames $l = 85$ to 160 of the *Campus* sequence.

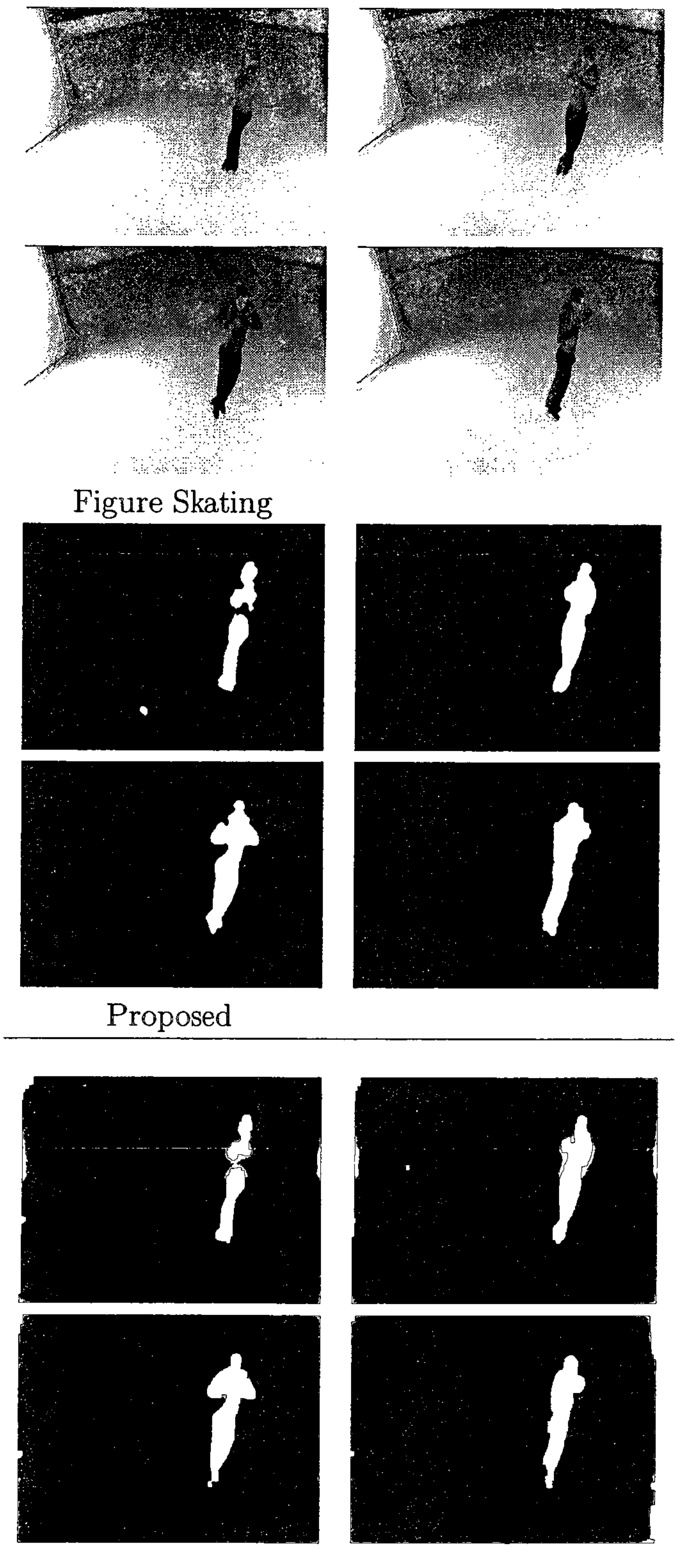


Figure A.5: Visual comparison of the segmentation output for frames $l = 10$ to 13 of the *Figure Skating* sequence between the proposed and reference [6] methods.

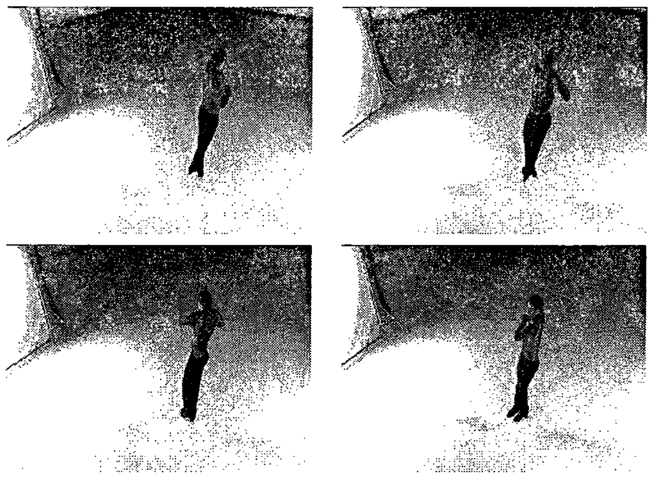
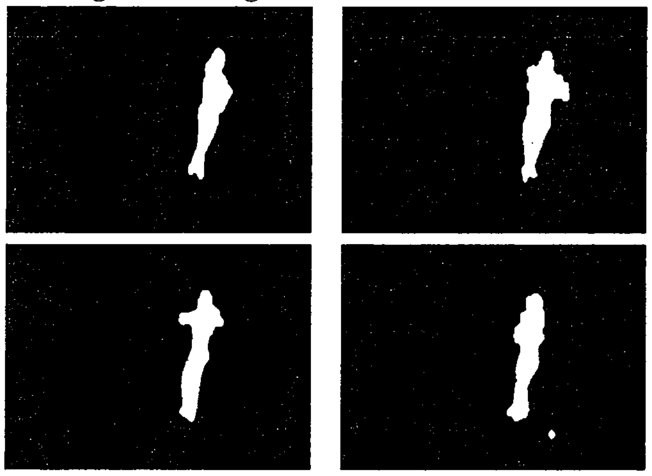


Figure Skating


Proposed

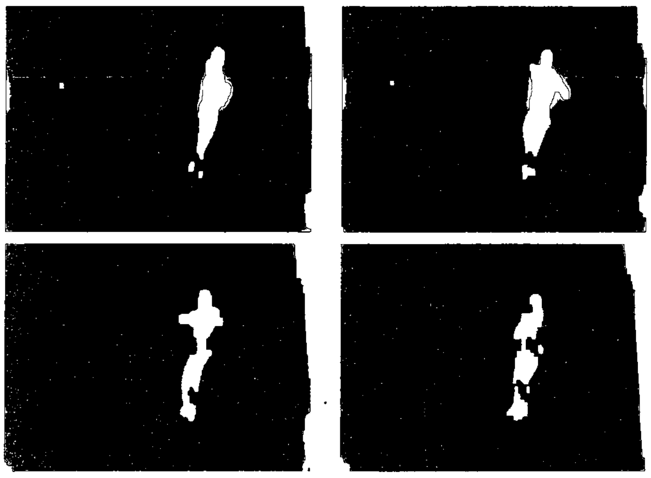

Reference

Figure A.6: Visual comparison of the segmentation output for frames $l = 14$ to 17 of the *Figure Skating* sequence between the proposed and reference [6] methods.

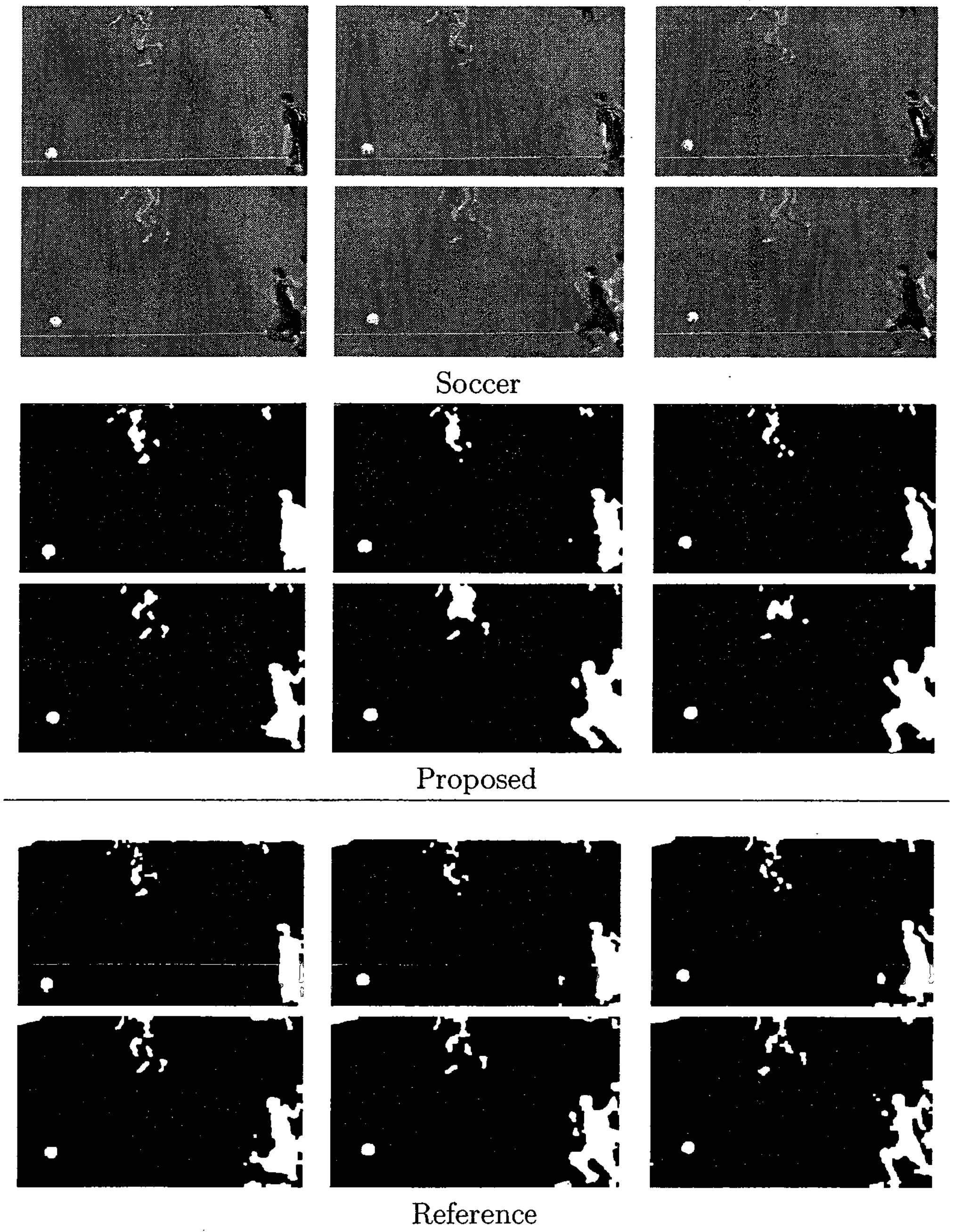


Figure A.7: Visual comparison of the segmentation output for frames $l = 20$ to 30 of the *Soccer* sequence between the proposed and reference [6] methods. To compensate for the captured video being in slow motion, we use $N_r = 20$ for this sequence.