MOVING OBJECT DETECTION, TRACKING AND CLASSIFICATION FOR SMART VIDEO SURVEILLANCE

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Yiğithan Dedeoğlu

August, 2004

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Uğur Güdükbay (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. A. Enis Çetin

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

MOVING OBJECT DETECTION, TRACKING AND CLASSIFICATION FOR SMART VIDEO SURVEILLANCE

Yiğithan Dedeoğlu
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Uğur Güdükbay
August, 2004

Video surveillance has long been in use to monitor security sensitive areas such as banks, department stores, highways, crowded public places and borders. The advance in computing power, availability of large-capacity storage devices and high speed network infrastructure paved the way for cheaper, multi sensor video surveillance systems. Traditionally, the video outputs are processed online by human operators and are usually saved to tapes for later use only after a forensic event. The increase in the number of cameras in ordinary surveillance systems overloaded both the human operators and the storage devices with high volumes of data and made it infeasible to ensure proper monitoring of sensitive areas for long times. In order to filter out redundant information generated by an array of cameras, and increase the response time to forensic events, assisting the human operators with identification of important events in video by the use of "smart" video surveillance systems has become a critical requirement. The making of video surveillance systems "smart" requires fast, reliable and robust algorithms for moving object detection, classification, tracking and activity analysis.

In this thesis, a smart visual surveillance system with real-time moving object detection, classification and tracking capabilities is presented. The system operates on both color and gray scale video imagery from a stationary camera. It can handle object detection in indoor and outdoor environments and under changing illumination conditions. The classification algorithm makes use of the shape of the detected objects and temporal tracking results to successfully categorize objects into pre-defined classes like human, human group and vehicle. The system is also able to detect the natural phenomenon fire in various scenes reliably. The proposed tracking algorithm successfully tracks video objects even in full occlusion cases. In addition to these, some important needs of a robust

smart video surveillance system such as removing shadows, detecting sudden illumination changes and distinguishing left/removed objects are met.

Keywords: Video-Based Smart Surveillance, Moving Object Detection, Background Subtraction, Object Tracking, Silhouette-Based Object Classification, Fire Detection.

ÖZET

AKILLI VİDEO GÖZETİMİ İÇİN HAREKETLİ NESNE BULMA, TAKİP ETME VE SINIFLANDIRMA

Yiğithan Dedeoğlu
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Uğur Güdükbay
Ağustos, 2004

Video gözetimi hassas güvenlik gerektiren banka, alışveriş merkezi, otoyol gibi kamuya açık kalabalık alanları izlemek için uzun süredir kullanılmaktadır. Bilgi işlem gücündeki artış, yüksek kapasiteli kayıt cihazlarının üretilmesi ve hızlı ağ altyapısı ucuz ve çok algılayıcılı sistemlerin üretilmesine ön ayak olmuştur. Elde edilen video görüntüsü canlı olarak operatörlerce izlenir ve daha sonra adli bir olayda kullanılmak üzere kaydedilir. Sıradan bir gözetim sisteminde bulunan kamera sayısındaki teknolojinin gelişmesine bağlı artış, hem operatörleri hem de kayıt cihazlarını aşırı hacimdeki bilgiye maruz bırakmış ve hasas güvenlik bölgelerinin uzun süreli video ile gözetimini verimsiz kılmıştır. Bir dizi kamera tarafından üretilen ve çoğunlukla gereksiz olan görüntü bilgisini elemek ve adli olaylara müdahale zamanını kısaltmak için operatörlere videodaki önemli olayları belirleyerek yardımcı olacak "akıllı" video gözetim sistemlerini geliştirmek kritik bir ihtiyaç haline gelmiştir. Video gözetim sistemlerini "akıllı" hale getirmek hızlı, güvenilir ve hatasız nesne bulma, sınıflandırma ve takip etme algoritmalarını gerektirmektedir.

Bu tezde, nesne bulma, sınıflandırma ve takip etme yeteneklerine sahip bir "akıllı" video gözetim sistemi sunulmuştur. Sistem sabit bir kameradan elde edilen renkli ve renksiz görüntüler üzerinde çalışabilmektedir. İç ve dış mekanlarda, değişen ışık koşulları altında çekilen video görüntülerinde yer alan nesneler bulunabilmektedir. Nesne sınıflandırma algoritması bulunan nesneleri şekillerinden ve nesne takip etme algoritmasından yararlanarak önceden tanımlanmış olan insan, insan grubu ve araç gibi sınıflara ayırabilmektedir. Önerilen sistem bina ve açık alan güvenliğinde çok önem arzeden yangını da güvenilir bir şekilde bulabilmektedir. Nesne takip algoritması başka nesneler tarafından perdelenen nesneleri de takip edebilmektedir. Tüm bu özelliklere ek

olarak video gözetim sistemlerinin önemli ihtiyacı olan gölge bölgelerinin silinmesi, ani ışık değişimlerinin algılanması ve bırakılan ya da alınan nesnelerinin bulunması da gerçekleştirebilmektedir.

Anahtar sözcükler: Akıllı Video Gözetimi, Hareketli Nesne Bulma, Arka Plan Kestirimi, Nesne Takip Etme, Siluete Dayalı Nesne Sınıflandırma, Ateş Bulma.

Acknowledgement

I would like to express my gratitude to Assist. Prof. Dr. Uğur Güdükbay for his supervision, encouragement, suggestions and trust throughout the development of this theses.

I owe special thanks to Prof. Dr. A. Enis Çetin for his guidance, support and invaluable discussions that encouraged me in my research. I am grateful to my thesis committee member Prof. Dr. Özgür Ulusoy for reading my thesis and helpful comments.

I am also thankful to Dr. M. Bilgay Akhan (CEO, visiOprime, UK) for supporting some parts of our research financially.

Without the support of my colleagues this research would not be that much enjoyable for me. I am especially grateful to Behçet Uğur Töreyin for his invaluable contributions, cooperation and discussions. I am also indebted to Alptekin Temizel (visiOprime, UK), Anıl Aksay and Erdem Dengel for their support and comments.

I also would like to thank Aslı for her love and morale support that made everything easier for me.

Finally, my biggest gratitude is to my family (Demirhan, Hacer, Tuğcihan Dedeoğlu and Hilmi, Öznur Çelik) for their endless love, support and trust in me. Without them I would never come up to this stage.

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Video surveillance systems have long been in use to monitor security sensitive areas. The history of video surveillance consists of three generations of systems which are called 1GSS, 2GSS and 3GSS [36].

The first generation surveillance systems (1GSS, 1960-1980) were based on analog sub systems for image acquisition, transmission and processing. They extended human eye in spatial sense by transmitting the outputs of several cameras monitoring a set of sites to the displays in a central control room. They had the major drawbacks like requiring high bandwidth, difficult archiving and retrieval of events due to large number of video tape requirements and difficult online event detection which only depended on human operators with limited attention span.

The next generation surveillance systems (2GSS, 1980-2000) were hybrids in the sense that they used both analog and digital sub systems to resolve some drawbacks of its predecessors. They made use of the early advances in digital video processing methods that provide assistance to the human operators by filtering out spurious events. Most of the work during 2GSS is focused on real-time event detection.

Third generation surveillance systems (3GSS, 2000-) provide end-to-end digital systems. Image acquisition and processing at the sensor level, communication

through mobile and fixed heterogeneous broadband networks and image storage at the central servers benefit from low cost digital infrastructure.

Unlike previous generations, in 3GSS some part of the image processing is distributed towards the sensor level by the use of intelligent cameras that are able to digitize and compress acquired analog image signals and perform image analysis algorithms like motion and face detection with the help of their attached digital computing components.

The ultimate goal of 3GSS is to allow video data to be used for online alarm generation to assist human operators and for offline inspection effectively. In order to achieve this goal, 3GSS will provide smart systems that are able to generate real-time alarms defined on complex events and handle distributed storage and content-based retrieval of video data.

The making of video surveillance systems "smart" requires fast, reliable and robust algorithms for moving object detection, classification, tracking and activity analysis. Starting from the 2GSS, a considerable amount of research has been devoted for the development of these intelligent algorithms.

Moving object detection is the basic step for further analysis of video. It handles segmentation of moving objects from stationary background objects. This not only creates a focus of attention for higher level processing but also decreases computation time considerably. Commonly used techniques for object detection are background subtraction, statistical models, temporal differencing and optical flow. Due to dynamic environmental conditions such as illumination changes, shadows and waving tree branches in the wind object segmentation is a difficult and significant problem that needs to be handled well for a robust visual surveillance system.

Object classification step categorizes detected objects into predefined classes such as human, vehicle, animal, clutter, etc. It is necessary to distinguish objects from each other in order to track and analyze their actions reliably. Currently, there are two major approaches towards moving object classification, which are shape-based and motion-based methods [49]. Shape-based methods make use of

the objects' 2D spatial information whereas motion-based methods use temporal tracked features of objects for the classification solution. Detecting natural phenomenon such as fire and smoke may be incorporated into object classification components of the visual surveillance systems. Detecting fire and raising alarms make the human operators take precautions in a shorter time which would save properties, forests and animals from catastrophic consequences.

The next step in the video analysis is tracking, which can be simply defined as the creation of temporal correspondence among detected objects from frame to frame. This procedure provides temporal identification of the segmented regions and generates cohesive information about the objects in the monitored area such as trajectory, speed and direction. The output produced by tracking step is generally used to support and enhance motion segmentation, object classification and higher level activity analysis.

The final step of the smart video surveillance systems is to recognize the behaviors of objects and create high-level semantic descriptions of their actions. It may simply be considered as a classification problem of the temporal activity signals of the objects according to pre-labeled reference signals representing typical human actions [49].

The outputs of these algorithms can be used both for providing the human operator with high level data to help him to make the decisions more accurately and in a shorter time and for offline indexing and searching stored video data effectively. The advances in the development of these algorithms would lead to breakthroughs in applications that use visual surveillance. Below are some scenarios that smart surveillance systems and algorithms might handle [10, 2, 16, 33, 15, 13, 45, 27, 41, 39, 49, 21]:

Public and commercial security:

- Monitoring of banks, department stores, airports, museums, stations, private properties and parking lots for crime prevention and detection
- Patrolling of highways and railways for accident detection

- Surveillance of properties and forests for fire detection
- Observation of the activities of elderly and infirm people for early alarms and measuring effectiveness of medical treatments
- Access control

Smart video data mining:

- Measuring traffic flow, pedestrian congestion and athletic performance
- Compiling consumer demographics in shopping centers and amusement parks
- Extracting statistics from sport activities
- Counting endangered species
- Logging routine maintenance tasks at nuclear and industrial facilities
- Artistic performance evaluation and self learning

Law enforcement:

- Measuring speed of vehicles
- Detecting red light crossings and unnecessary lane occupation

Military security:

- Patrolling national borders
- Measuring flow of refugees
- Monitoring peace treaties
- Providing secure regions around bases

- Assisting battlefield command and control

The use of smart object detection, tracking and classification algorithms are not limited to video surveillance only. Other application domains also benefit from the advances in the research on these algorithms. Some examples are virtual reality, video compression, human machine interface, augmented reality, video editing and multimedia databases.

1.1 Overview

In this thesis, we present a smart visual surveillance system with real-time moving object detection, classification and tracking capabilities. The system operates on both color and gray scale video imagery from a stationary camera.

In the proposed system moving object detection is handled by the use of an adaptive background subtraction scheme [10] which reliably works in indoor and outdoor environments. We also present two other object detection schemes, temporal differencing [10] and adaptive background mixture models [44], for performance and detection quality comparison.

In adaptive background subtraction method, a reference background is initialized at the start of the system with the first few frames of video and updated to adapt to short and long term dynamic scene changes during the operational period. At each new frame, foreground pixels are detected by subtracting the intensity values from the background and filtering the absolute value of the differences with a dynamic threshold per pixel. The reference background and the threshold values are updated by using the foreground pixel information. The detected foreground pixels usually contain noise due to image acquisition errors, small movements like tree leaves, reflections and foreground objects with textures colored similar to the background. These isolated pixels are filtered by the use of a sequence of morphological operations dilation and erosion. After this step, the individual pixels are grouped and labeled by using a two pass component labeling algorithm to create connected moving regions. These regions are further

processed to group disconnected blobs and to eliminate relatively small sized regions. After grouping, each detected foreground object is represented with its bounding box, area, center of mass and color histogram which will be used in later steps.

After segmenting moving pixels from the static background of the scene, connected regions are classified into predetermined object categories human, human group and vehicle. The classification algorithm depends on the comparison of the silhouettes of the detected objects with pre-labeled (classified) templates in an object silhouette database. The template database is created by collecting sample object silhouettes from sample videos and labeling them manually with appropriate categories. The silhouettes of the objects are extracted from the connected foreground regions by using a contour tracing algorithm [19]. Next, the distance between each boundary pixel and the center of mass point is calculated to create a distance signal starting from the top pixel and continuing clock-wise until reaching the same pixel. The distance signals are first normalized to be of the same length, then smoothed and finally normalized again to cover the same area. The comparison metric used in matching the templates with the detected objects are the L_1 distance [42] of normalized silhouette distance signals. The class of the template silhouette with minimum distance from the detected object's silhouette is assigned to the object's class. Temporal tracking information is used to support classification decision.

Detecting the natural phenomenon fire besides normal object motion would be an advantage of a visual surveillance system, thus, the presented system is able to detect fire in indoor and outdoor environments. Conventional point smoke and fire detectors typically detect the presence of certain particles generated by smoke and fire by ionization or photometry. An important weakness of point detectors is that they are distance limited and fail in open or large spaces. The strength of using video in fire detection is the ability to serve large and open spaces. Current fire and flame detection algorithms are based on the use of color and simple motion information in video [27]. In addition to detecting fire and flame colored moving regions, the method presented in this thesis analyzes the motion patterns, the temporal periodicity and spatial variance of high-frequency

behavior extensively.

As the final step in the presented system, the tracking algorithm tracks the detected objects in successive frames by using a correspondence-based matching scheme. It also handles multi-occlusion cases where some objects might be fully occluded by others. It uses 2D object features such as position and size to match corresponding objects from frame to frame. It keeps color histograms of detected objects in order resolve object identities after a split of an occlusion group. The output of the tracking step supports both motion segmentation and object classification steps.

1.2 Motivation

Understanding activities of objects moving in a scene by the use of video is both a challenging scientific problem and a very fertile domain with many promising applications. Thus, it draws attentions of several researchers, institutions and commercial companies [49]. Our motivation in studying this problem is to create a visual surveillance system with real-time moving object detection, classification, tracking and activity analysis capabilities. The presented system handles all of the above methods except activity recognition which will likely be the future step of our research.

1.3 Organization of the Thesis

The remaining part of this thesis is organized as follows. Chapter 2 presents a brief survey in moving object detection, tracking and classification for video surveillance applications. Our methods for moving object detection and tracking are explained in Chapter 3. Our novel object classification method is presented in Chapter 4. In the next chapter we explain the fire detection approach. Experimental results of the proposed system are presented in Chapter 6. Finally, Chapter 7 concludes the thesis with the suggestions for future research.

Chapter 2

A Survey in Smart Video Surveillance

There have been a number of surveys about object detection, classification, tracking and activity analysis in the literature [13, 1, 49]. The survey we present here covers only those work that are in the same context as our study. However, for comprehensive completeness, we also give brief information on some techniques which are used for similar tasks that are not covered in our study.

A generic video processing framework for smart algorithms is shown in Figure 2.1. Although, some steps require interchange of information with other levels, this framework provides a good structure for the discussion throughout this survey.

2.1 Moving Object Detection

Each application that benefit from smart video processing has different needs, thus requires different treatment. However, they have something in common: moving objects. Thus, detecting regions that correspond to moving objects such as people and vehicles in video is the first basic step of almost every vision system

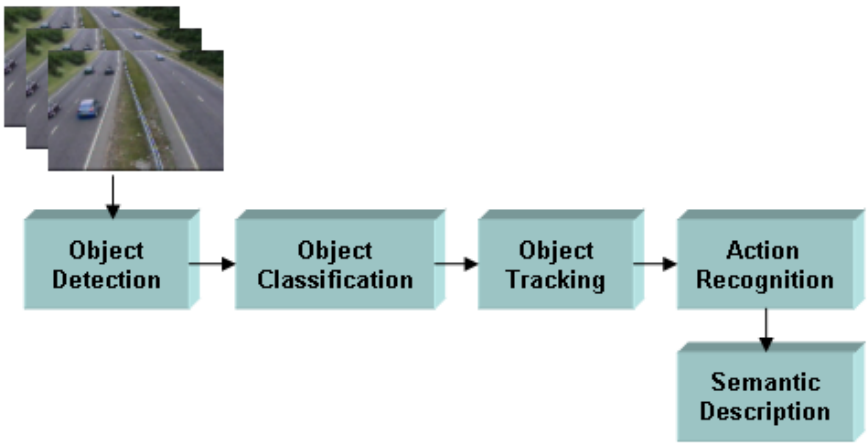


Figure 2.1: A generic framework for smart video processing algorithms.

since it provides a focus of attention and simplifies the processing on subsequent analysis steps. Due to dynamic changes in natural scenes such as sudden illumination and weather changes, repetitive motions that cause clutter (tree leaves moving in blowing wind), motion detection is a difficult problem to process reliably. Frequently used techniques for moving object detection are background subtraction, statistical methods, temporal differencing and optical flow whose descriptions are given below.

2.1.1 Background Subtraction

Background subtraction is particularly a commonly used technique for motion segmentation in static scenes [34]. It attempts to detect moving regions by subtracting the current image pixel-by-pixel from a reference background image that is created by averaging images over time in an initialization period. The pixels where the difference is above a threshold are classified as foreground. After creating a foreground pixel map, some morphological post processing operations such as erosion, dilation and closing are performed to reduce the effects of noise and enhance the detected regions. The reference background is updated with new images over time to adapt to dynamic scene changes.

There are different approaches to this basic scheme of background subtraction in terms of foreground region detection, background maintenance and post processing.

In [20] Heikkila and Silven uses the simple version of this scheme where a pixel at location (x, y) in the current image I_t is marked as foreground if

$$|I_t(x, y) - B_t(x, y)| > \tau \tag{2.1}$$

is satisfied where τ is a predefined threshold. The background image B_T is updated by the use of an Infinite Impulse Response (IIR) filter as follows:

$$B_{t+1} = \alpha I_t + (1 - \alpha) B_t \tag{2.2}$$

The foreground pixel map creation is followed by morphological closing and the elimination of small-sized regions.

Although background subtraction techniques perform well at extracting most of the relevant pixels of moving regions even they stop, they are usually sensitive to dynamic changes when, for instance, stationary objects uncover the background (e.g. a parked car moves out of the parking lot) or sudden illumination changes occur.

2.1.2 Statistical Methods

More advanced methods that make use of the statistical characteristics of individual pixels have been developed to overcome the shortcomings of basic background subtraction methods. These statistical methods are mainly inspired by the background subtraction methods in terms of keeping and dynamically updating statistics of the pixels that belong to the background image process. Foreground pixels are identified by comparing each pixel's statistics with that of the background model. This approach is becoming more popular due to its reliability in scenes that contain noise, illumination changes and shadow [49].

The W4 [17] system uses a statistical background model where each pixel is

represented with its minimum (M) and maximum (N) intensity values and maximum intensity difference (D) between any consecutive frames observed during initial training period where the scene contains no moving objects. A pixel in the current image I_t is classified as foreground if it satisfies:

$$|M(x,y) - I_t(x,y)| > D(x,y) \quad or \quad |N(x,y) - I_t(x,y)| > D(x,y) \tag{2.3}$$

After thresholding, a single iteration of morphological erosion is applied to the detected foreground pixels to remove one-pixel thick noise. In order to grow the eroded regions to their original sizes, a sequence of erosion and dilation is performed on the foreground pixel map. Also, small-sized regions are eliminated after applying connected component labeling to find the regions. The statistics of the background pixels that belong to the non-moving regions of current image are updated with new image data.

As another example of statistical methods, Stauffer and Grimson [44] described an adaptive background mixture model for real-time tracking. In their work, every pixel is separately modeled by a mixture of Gaussians which are updated online by incoming image data. In order to detect whether a pixel belongs to a foreground or background process, the Gaussian distributions of the mixture model for that pixel are evaluated. An implementation of this model is used in our system and its details are explained in Section 3.1.1.2.

2.1.3 Temporal Differencing

Temporal differencing attempts to detect moving regions by making use of the pixel-by-pixel difference of consecutive frames (two or three) in a video sequence. This method is highly adaptive to dynamic scene changes, however, it generally fails in detecting whole relevant pixels of some types of moving objects. A sample object for inaccurate motion detection is shown in Figure 2.2. The mono colored region of the human on the left hand side makes the temporal differencing algorithm to fail in extracting all pixels of the human's moving region. Also, this method fails to detect stopped objects in the scene. Additional methods need to be adopted in order to detect stopped objects for the success of higher level

(a) (b)

Figure 2.2: Temporal differencing sample. (a) A sample scene with two moving objects. (b) Temporal differencing fails to detect all moving pixels of the object on the left hand side since it is uniform colored. The detected moving regions are marked with red pixels.

processing.

Lipton et al. presented a two-frame differencing scheme where the pixels that satisfy the following equation are marked as foreground [29].

$$|I_t(x, y) - I_{t-1}(x, y)| > \tau \tag{2.4}$$

In order to overcome shortcomings of two frame differencing in some cases, three frame differencing can be used [49]. For instance, Collins et al. developed a hybrid method that combines three-frame differencing with an adaptive background subtraction model for their VSAM project [10]. The hybrid algorithm successfully segments moving regions in video without the defects of temporal differencing and background subtraction.

2.1.4 Optical Flow

Optical flow methods make use of the flow vectors of moving objects over time to detect moving regions in an image. They can detect motion in video sequences even from a moving camera, however, most of the optical flow methods

are computationally complex and cannot be used real-time without specialized hardware [49].

2.1.5 Shadow and Light Change Detection

The algorithms described above for motion detection perform well on indoor and outdoor environments and have been used for real-time surveillance for years. However, without special care, most of these algorithms are susceptible to both local (e.g. shadows and highlights) and global illumination changes (e.g. sun being covered/uncovered by clouds). Shadows cause the motion detection methods fail in segmenting only the moving objects and make the upper levels such as object classification to perform inaccurate. The proposed methods in the literature mostly use either chromaticity [21, 35, 6, 53, 26] or stereo [15] information to cope with shadows and sudden light changes.

Horprasert et al. present a novel background subtraction and shadow detection method [21]. In their method, each pixel is represented by a color model that separates brightness from the chromaticity component. A given pixel is classified into four different categories (background, shaded background or shadow, highlighted background and moving foreground object) by calculating the distortion of brightness and chromaticity between the background and the current image pixels. Like [21], the approach described by McKenna et al. in [35] uses chromaticity and gradient information to cope with shadows. They make use of the observation that an area cast into shadow results in significant change in intensity without much change in chromaticity. They also use the gradient information in moving regions to ensure reliability of their method in ambiguous cases.

The method presented in [6] adopts a shadow detection scheme which depends on two heuristics: a) pixel intensity values within shadow regions tend to decrease in most cases when compared to the background image, b) the intensity reduction rate changes smoothly between neighboring pixels and most shadow edges do not have strong edges.

An efficient method to deal with shadows is using stereo as presented in W4S [15] system. In W4S, stereo image is generated by an inexpensive real-time device called SVM which uses two or more images to calculate a range image by using simple stereo image geometry. With the help of the range information provided by SVM, W4S is able to cope with shadows, sudden illumination changes and complex occlusion cases.

In some systems, a global light change is detected by counting the number of foreground pixels and if the total number exceeds some threshold (e.g. 50% of the total image size), the system is reset to adapt to the sudden illumination change [37, 55].

2.2 Object Classification

Moving regions detected in video may correspond to different objects in real-world such as pedestrians, vehicles, clutter, etc. It is very important to recognize the type of a detected object in order to track it reliably and analyze its activities correctly. Currently, there are two major approaches towards moving object classification which are shape-based and motion-based methods [49]. Shape-based methods make use of the objects' 2D spatial information whereas motion-based methods use temporally tracked features of objects for the classification solution.

2.2.1 Shape-based Classification

Common features used in shape-based classification schemes are the bounding rectangle, area, silhouette and gradient of detected object regions.

The approach presented in [29] makes use of the objects' silhouette contour length and area information to classify detected objects into three groups: human, vehicle and other. The method depends on the assumption that humans are, in general, smaller than vehicles and have complex shapes. Dispersedness is used as the classification metric and it is defined in terms of object's area and contour

length (perimeter) as follows:

$$Dispersedness = \frac{Perimeter^2}{Area} \tag{2.5}$$

Classification is performed at each frame and tracking results are used to improve temporal classification consistency.

The classification method developed by Collins et al. [10] uses view dependent visual features of detected objects to train a neural network classifier to recognize four classes: human, human group, vehicle and clutter. The inputs to the neural network are the dispersedness, area and aspect ratio of the object region and the camera zoom magnification. Like the previous method, classification is performed at each frame and results are kept in a histogram to improve temporal consistency of classification.

Saptharishi et al. propose a classification scheme which uses a logistic linear neural network trained with Differential Learning to recognize two classes: vehicle and people [41]. Papageorgiou et al. presents a method that makes use of the Support Vector Machine classification trained by wavelet transformed object features (edges) in video images from a sample pedestrian database [38]. This method is used to recognize moving regions that correspond to humans.

Another classification method proposed by Brodsky et al. [11] uses a Radial Basis Function (RBF) classifier which has a similar architecture like a three-layer back-propagation network. The input to the classifier is the normalized gradient image of the detected object regions.

2.2.2 Motion-based Classification

Some of the methods in the literature use only temporal motion features of objects in order to recognize their classes [8, 51, 28]. In general, they are used to distinguish non-rigid objects (e.g. human) from rigid objects (e.g. vehicles). The method proposed in [8] is based on the temporal self-similarity of a moving object. As an object that exhibits periodic motion evolves, its self-similarity measure also shows a periodic motion. The method exploits this clue to categorize

moving objects using periodicity.

Optical flow analysis is also useful to distinguish rigid and non-rigid objects. A. J. Lipton proposed a method that makes use of the local optical flow analysis of the detected object regions [28]. It is expected that non-rigid objects such as humans will present high average residual flow whereas rigid objects such as vehicles will present little residual flow. Also, the residual flow generated by human motion will have a periodicity. By using this cue, human motion, thus humans, can be distinguished from other objects such as vehicles.

2.3 Fire Detection

The number of papers that discuss fire detection using video is very few in computer vision literature. Most of the proposed methods exploit the color and motion features of fire.

Healey et al. [18] use a model which is based only on color characteristics of fire. Obviously this method generates false alarms due to fire colored regions. An improved approach which makes use of motion information as well as the color property is presented by Philips et al. [23].

Recently, Liu and Ahuja [30] presented a method that defines spectral, spatial and temporal models of fire to detect its presence in video. The spectral model is represented in terms of fire pixel color probability density. The spatial model describes the spatial structure of a fire region and the temporal model captures the changes in the spatial structure over time.

2.4 Object Tracking

Tracking is a significant and difficult problem that arouses interest among computer vision researchers. The objective of tracking is to establish correspondence of objects and object parts between consecutive frames of video. It is a significant

task in most of the surveillance applications since it provides cohesive temporal data about moving objects which are used both to enhance lower level processing such as motion segmentation and to enable higher level data extraction such as activity analysis and behavior recognition. Tracking has been a difficult task to apply in congested situations due to inaccurate segmentation of objects. Common problems of erroneous segmentation are long shadows, partial and full occlusion of objects with each other and with stationary items in the scene. Thus, dealing with shadows at motion detection level and coping with occlusions both at segmentation level and at tracking level is important for robust tracking.

Tracking in video can be categorized according to the needs of the applications it is used in or according to the methods used for its solution. Whole body tracking is generally adequate for outdoor video surveillance whereas objects' part tracking is necessary for some indoor surveillance and higher level behavior understanding applications.

There are two common approaches in tracking objects as a whole [2]: one is based on correspondence matching and other one carries out explicit tracking by making use of position prediction or motion estimation. On the other hand, the methods that track parts of objects (generally humans) employ model-based schemes to locate and track body parts. Some example models are stick figure, Cardboard Model [25], 2D contour and 3D volumetric models.

W4 [17] combines motion estimation methods with correspondence matching to track objects. It is also able to track parts of people such as heads, hands, torso and feet by using the Cardboard Model [25] which represents relative positions and sizes of body parts. It keeps appearance templates of individual objects to handle matching even in merge and split cases.

Amer [2] presents a non-linear voting based scheme for tracking objects as a whole. It integrates object features like size, shape, center of mass and motion by voting and decides final matching with object correspondence. This method can also detect object split and fusion and handle occlusions.

Stauffer et al. [45] employs a linearly predictive multiple hypotheses tracking

algorithm. The algorithm incorporates size and positions of objects for seeding and maintaining a set of Kalman filters for motion estimation. Also, Extended Kalman filters are used for trajectory prediction and occlusion handling in the work of Rosales and Sclaroff [40].

As an example of model based body part tracking system, Pfinder [52] makes use of a multi-class statistical model of color and shape to track head and hands of people in real-time.

Chapter 3

Object Detection and Tracking

The overview of our real time video object detection, classification and tracking system is shown in Figure 3.1. The proposed system is able to distinguish transitory and stopped foreground objects from static background objects in dynamic scenes; detect and distinguish left and removed objects; classify detected objects into different groups such as human, human group and vehicle; track objects and generate trajectory information even in multi-occlusion cases and detect fire in video imagery. In this and following chapters we describe the computational models employed in our approach to reach the goals specified above.

Our system is assumed to work real time as a part of a video-based surveillance system. The computational complexity and even the constant factors of the algorithms we use are important for real time performance. Hence, our decisions on selecting the computer vision algorithms for various problems are affected by their computational run time performance as well as quality. Furthermore, our system's use is limited only to stationary cameras and video inputs from Pan/Tilt/Zoom cameras where the view frustum may change arbitrarily are not supported.

The system is initialized by feeding video imagery from a static camera monitoring a site. Most of the methods are able to work on both color and monochrome video imagery. The first step of our approach is distinguishing foreground objects

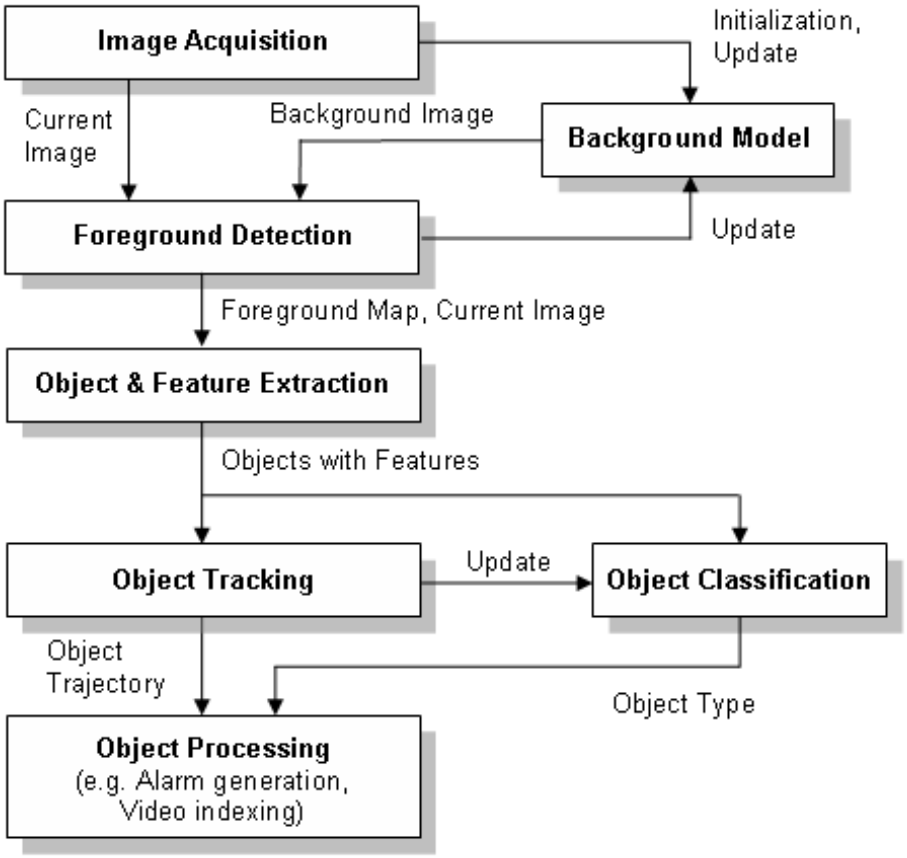


Figure 3.1: The system block diagram.

from stationary background. To achieve this, we use a combination of adaptive background subtraction and low-level image post-processing methods to create a foreground pixel map at every frame. We then group the connected regions in the foreground map to extract individual object features such as bounding box, area, center of mass and color histogram.

Our novel object classification algorithm makes use of the foreground pixel map belonging to each individual connected region to create a silhouette for the object. The silhouette and center of mass of an object are used to generate a distance signal. This signal is scaled, normalized and compared with pre-labeled signals in a template database to decide on the type of the object. The output of the tracking step is used to attain temporal consistency in the classification step.

The object tracking algorithm utilizes extracted object features together with a correspondence matching scheme to track objects from frame to frame. The color histogram of an object produced in previous step is used to match the correspondences of objects after an occlusion event. The output of the tracking step is object trajectory information which is used to calculate direction and speed of the objects in the scene.

After gathering information on objects' features such as type, trajectory, size and speed various high level processing can be applied on these data. A possible use is real-time alarm generation by pre-defining event predicates such as "A *human* moving in direction d at speed more than s causes alarm a_1." or "A *vehicle* staying at location l more than t seconds causes alarm a_2.". Another opportunity we may make use of the produced video object data is to create an index on stored video data for offline smart search. Both alarm generation and video indexing are critical requirements of a visual surveillance system to increase response time to forensic events.

The remainder of this chapter presents the computational models and methods we adopted for object detection and tracking. Our object classification approach is explained in the next chapter.

3.1 Object Detection

Distinguishing foreground objects from the stationary background is both a significant and difficult research problem. Almost all of the visual surveillance systems' first step is detecting foreground objects. This both creates a focus of attention for higher processing levels such as tracking, classification and behavior understanding and reduces computation time considerably since only pixels belonging to foreground objects need to be dealt with. Short and long term dynamic scene changes such as repetitive motions (e. g. waiving tree leaves), light reflectance, shadows, camera noise and sudden illumination variations make reliable and fast object detection difficult. Hence, it is important to pay necessary attention to object detection step to have reliable, robust and fast visual surveillance system.

The system diagram of our object detection method is shown in Figure 3.2. Our method depends on a six stage process to extract objects with their features in video imagery. The first step is the background scene initialization. There are various techniques used to model the background scene in the literature (see Section 2.1). In order to evaluate the quality of different background scene models for object detection and to compare run-time performance, we implemented three of these models which are adaptive background subtraction, temporal frame differencing and adaptive online Gaussian mixture model. The background scene related parts of the system is isolated and its coupling with other modules is kept minimum to let the whole detection system to work flexibly with any one of the background models.

Next step in the detection method is detecting the foreground pixels by using the background model and the current image from video. This pixel-level detection process is dependent on the background model in use and it is used to update the background model to adapt to dynamic scene changes. Also, due to camera noise or environmental effects the detected foreground pixel map contains noise. Pixel-level post-processing operations are performed to remove noise in the foreground pixels.

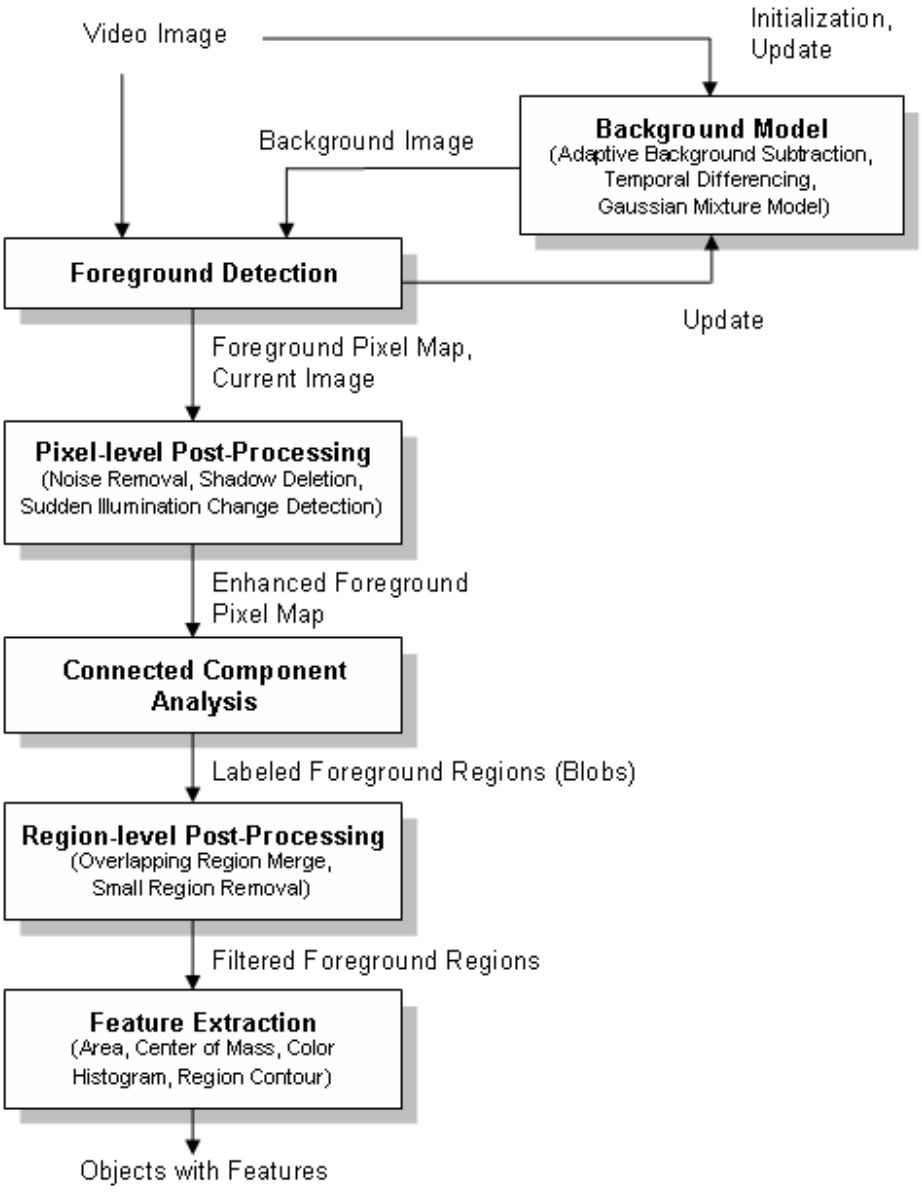


Figure 3.2: The object detection system diagram.

Once we get the filtered foreground pixels, in the next step, connected regions are found by using a connected component labeling algorithm and objects' bounding rectangles are calculated. The labeled regions may contain near but disjoint regions due to defects in foreground segmentation process. Hence, it is experimentally found to be effective to merge those overlapping isolated regions. Also, some relatively small regions caused by environmental noise are eliminated in the region-level post-processing step.

In the final step of the detection process, a number of object features are extracted from current image by using the foreground pixel map. These features are the area, center of mass and color histogram of the regions corresponding to objects.

3.1.1 Foreground Detection

We use a combination of a background model and low-level image post-processing methods to create a foreground pixel map and extract object features at every video frame. Background models generally have two distinct stages in their process: initialization and update. Following sections describe the initialization and update mechanisms together with foreground region detection methods used in the three background models we tested in our system. The experimental comparison of the computational run-time and detection qualities of these models are given in Section 6.2.

3.1.1.1 Adaptive Background Subtraction Model

Our implementation of background subtraction algorithm is partially inspired by the study presented in [10] and works on grayscale video imagery from a static camera. Our background subtraction method initializes a reference background with the first few frames of video input. Then it subtracts the intensity value of each pixel in the current image from the corresponding value in the reference background image. The difference is filtered with an adaptive threshold per pixel

to account for frequently changing noisy pixels. The reference background image and the threshold values are updated with an IIR filter to adapt to dynamic scene changes.

Let $I_n(x)$ represent the gray-level intensity value at pixel position (x) and at time instance n of video image sequence I which is in the range [0, 255]. Let $B_n(x)$ be the corresponding background intensity value for pixel position (x) estimated over time from video images I_0 through I_{n-1}. As the generic background subtraction scheme suggests, a pixel at position (x) in the current video image belongs to foreground if it satisfies:

$$|I_n(x) - B_n(x)| > T_n(x) \tag{3.1}$$

where $T_n(x)$ is an adaptive threshold value estimated using the image sequence I_0 through I_{n-1}. The Equation 3.1 is used to generate the foreground pixel map which represents the foreground regions as a binary array where a 1 corresponds to a foreground pixel and a 0 stands for a background pixel.

The reference background $B_n(x)$ is initialized with the first video image I_0, $B_0 = I_0$, and the threshold image is initialized with some pre-determined value (e.g. 15).

Since our system will be used in outdoor environments as well as indoor environments, the background model needs to adapt itself to the dynamic changes such as global illumination change (day night transition) and long term background update (parking a car in front of a building). Therefore the reference background and threshold images are dynamically updated with incoming images. The update scheme is different for pixel positions which are detected as belonging to foreground ($x \in FG$) and which are detected as part of the background ($x \in BG$):

$$B_{n+1}(x) = \begin{cases} \alpha B_n(x) + (1-\alpha) I_n(x), & x \in BG \\ \beta B_n(x) + (1-\beta) I_n(x), & x \in FG \end{cases} \tag{3.2}$$

$$T_{n+1}(x) = \begin{cases} \alpha T_n(x) + (1-\alpha)(\gamma \times |I_n(x) - B_n(x)|), & x \in BG \\ T_n(x), & x \in FG \end{cases} \tag{3.3}$$

where α, β ($\in$ [0.0, 1.0]) are learning constants which specify how much information from the incoming image is put to the background and threshold images. In other words, if each background pixel is considered as a time series, the background image is a weighted local temporal average of the incoming image sequence and the threshold image is a weighted local temporal average of γ times the difference of incoming images and the background. The values for α, β and γ are experimentally determined by examining several indoor and outdoor video clips.

Our update mechanism for background is different than traditional background update and the one presented in [10] since we update the background for all types of pixels ($x \in FG$ or $x \in BG$). In typical background subtraction methods the reference background image is updated only for pixels belonging to background ($x \in BG$). This would allow them to adapt to repetitive noise and avoid merging moving objects into the scene to the background. However, in order to diffuse long term scene changes to the background, the regions in the background corresponding to the foreground object regions need also be updated.

The subtle point in this update is choosing the correct value for β. If it is too small, foreground objects will be merged to the reference background soon and it will lead to inaccurate segmentation in later frames. Also, detecting stopped objects will not be possible. If it is too big, objects may never be diffused into the background image, thus background model would not adapt to long-term scene changes. In the extreme case where $\beta = 1.0$, the Equation 3.2 is equivalent to the background update scheme presented in [10].

A sample foreground region detection is shown in Figure 3.3. The first image is the estimated reference background of the monitored site. The second image is captured at a later step and contains two foreground objects (two people). The third image shows the detected foreground pixel map using background subtraction.

(a) (b) (c)

Figure 3.3: Adaptive Background Subtraction sample. (a) Estimated background (b) Current image (c) Detected region

3.1.1.2 Adaptive Gaussian Mixture Model

Stauffer and Grimson [44] presented a novel adaptive online background mixture model that can robustly deal with lighting changes, repetitive motions, clutter, introducing or removing objects from the scene and slowly moving objects. Their motivation was that a unimodal background model could not handle image acquisition noise, light change and multiple surfaces for a particular pixel at the same time. Thus, they used a mixture of Gaussian distributions to represent each pixel in the model. Due to its promising features, we implemented and integrated this model in our visual surveillance system.

In this model, the values of an individual pixel (e. g. scalars for gray values or vectors for color images) over time is considered as a "pixel process" and the recent history of each pixel, $\{X_1, \ldots, X_t\}$, is modeled by a mixture of K Gaussian distributions. The probability of observing current pixel value then becomes:

$$P(X_t) = \sum_{i=1}^{K} w_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{3.4}$$

where $w_{i,t}$ is an estimate of the weight (what portion of the data is accounted for this Gaussian) of the i^{th} Gaussian ($G_{i,t}$) in the mixture at time t, $\mu_{i,t}$ is the mean value of $G_{i,t}$ and $\Sigma_{i,t}$ is the covariance matrix of $G_{i,t}$ and η is a Gaussian probability density function:

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1} (X_t-\mu_t)} \tag{3.5}$$

Decision on K depends on the available memory and computational power. Also, the covariance matrix is assumed to be of the following form for computational efficiency:

$$\Sigma_{k,t} = \alpha_k^2 \mathbf{I} \tag{3.6}$$

which assumes that red, green, blue color components are independent and have the same variance.

The procedure for detecting foreground pixels is as follows. At the beginning of the system, the K Gaussian distributions for a pixel are initialized with predefined mean, high variance and low prior weight. When a new pixel is observed in the image sequence, to determine its type, its RGB vector is checked against the K Gaussians, until a match is found. A match is defined as a pixel value within γ (=2.5) standard deviation of a distribution. Next, the prior weights of the K distributions at time t, $w_{k,t}$, are updated as follows:

$$w_{k,t} = (1 - \alpha) w_{k,t-1} + \alpha(M_{k,t}) \tag{3.7}$$

where α is the learning rate and $M_{k,t}$ is 1 for the matching Gaussian distribution and 0 for the remaining distributions. After this step the prior weights of the distributions are normalized and the parameters of the matching Gaussian are updated with the new observation as follows:

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho(X_t) \tag{3.8}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{3.9}$$

where

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k) \tag{3.10}$$

If no match is found for the new observed pixel, the Gaussian distribution with the least probability is replace with a new distribution with the current pixel value as its mean value, an initially high variance and low prior weight.

In order to detect the type (foreground or background) of the new pixel, the K Gaussian distributions are sorted by the value of w/σ. This ordered list of distributions reflect the most probable backgrounds from top to bottom since by

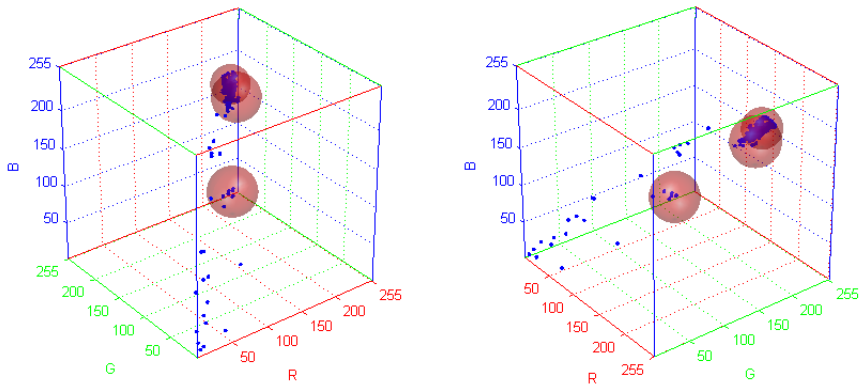


Figure 3.4: Two different views of a sample pixel processes (in blue) and corresponding Gaussian Distributions shown as alpha blended red spheres.

Equation 3.7 background pixel processes make the corresponding Gaussian distribution have larger prior weight and less variance. Then the first B distributions are chosen as the background model, where

$$B = argmin_b \left(\sum_{k=1}^{b} w_k > T \right) \tag{3.11}$$

and T is the minimum portion of the pixel data that should be accounted for by the background. If a small value is chosen for T, the background is generally unimodal. Figure 3.4 shows sample pixel processes and the Gaussian distributions as spheres covering these processes. The accumulated pixels define the background Gaussian distribution whereas scattered pixels are classified as foreground.

3.1.1.3 Temporal Differencing

Temporal differencing makes use of the pixel-wise difference between two or three consecutive frames in video imagery to extract moving regions. It is a highly adaptive approach to dynamic scene changes; however, it fails in extracting all relevant pixels of a foreground object especially when the object has uniform texture or moves slowly. When a foreground object stops moving, temporal differencing method fails in detecting a change between consecutive frames and loses

the object. Special supportive algorithms are required to detect stopped objects.

We implemented a two-frame temporal differencing method in our system. Let $I_n(x)$ represent the gray-level intensity value at pixel position (x) and at time instance n of video image sequence I which is in the range [0, 255]. The two-frame temporal differencing scheme suggests that a pixel is moving if it satisfies the following:

$$|I_n(x) - I_{n-1}(x)| > T_n(x) \tag{3.12}$$

Hence, if an object has uniform colored regions, the Equation 3.12 fails to detect some of the pixels inside these regions even if the object moves. The per-pixel threshold, T, is initially set to a pre-determined value and later updated as follows:

$$T_{n+1}(x) = \begin{cases} \alpha T_n(x) + (1-\alpha)(\gamma \times |I_n(x) - I_{n-1}(x)|), & x \in BG \\ T_n(x), & x \in FG \end{cases} \tag{3.13}$$

The implementation of two-frame differencing can be accomplished by exploiting the background subtraction method's model update parameters shown in Equation 3.2. If α and β are set to zero, the background holds the image I_{n-1} and background subtraction scheme becomes identical to two-frame differencing.

3.1.2 Pixel Level Post-Processing

The outputs of foreground region detection algorithms we explained in previous three sections generally contain noise and therefore are not appropriate for further processing without special post-processing. There are various factors that cause the noise in foreground detection such as:

- **Camera noise**: This is the noise caused by the camera's image acquisition components. The intensity of a pixel that corresponds to an edge between two different colored objects in the scene may be set to one of the object's color in one frame and to the other's color in the next frame.
- **Reflectance noise**: When a source of light, for instance sun, moves it makes some parts in the background scene to reflect light. This phenomenon

makes the foreground detection algorithms fail and detect reflectance as foreground regions.

- **Background colored object noise**: Some parts of the objects may have the same color as the reference background behind them. This resemblance causes some of the algorithms to detect the corresponding pixels as non-foreground and objects to be segmented inaccurately.
- **Shadows and sudden illumination change**: Shadows cast on objects are detected as foreground by most of the detection algorithms. Also, sudden illumination changes (e.g. turning on lights in a monitored room) makes the algorithms fail to detect actual foreground objects accurately.

Morphological operations, erosion and dilation[19], are applied to the foreground pixel map in order to remove noise that is caused by the first three of the items listed above. Our aim in applying these operations is removing noisy foreground pixels that do not correspond to actual foreground regions (let us name them non-foreground noise, shortly NFN) and to remove the noisy background pixels (non-background noise, shortly NBN) near and inside object regions that are actually foreground pixels. Erosion, as its name implies, erodes one-unit thick boundary pixels of foreground regions. Dilation is the reverse of erosion and expands the foreground region boundaries with one-unit thick pixels. The subtle point in applying these morphological filters is deciding on the order and amounts of these operations. The order of these operations affects the quality and the amount affects both the quality and the computational complexity of noise removal.

For instance, if we apply dilation followed by erosion we cannot get rid of one-pixel thick isolated noise regions (NFN) since the dilation operation would expand their boundaries with one pixel and the erosion will remove these extra pixels leaving the original noisy pixels. On the other hand, this order would successfully eliminate some of the non-background noise inside object regions. In case we apply these operations in reverse order, which is erosion followed by dilation, we would eliminate (NFN) regions but this time we would not be able to close holes inside objects (NBN).

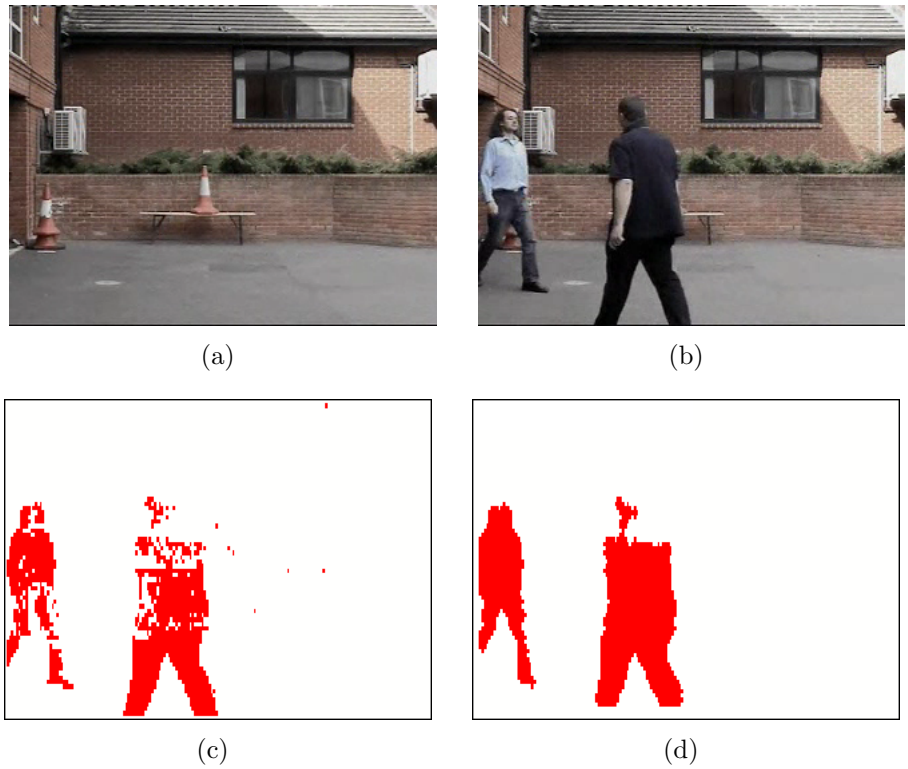


Figure 3.5: Pixel level noise removal sample. (a) Estimated background image (b) Current image (c) Detected foreground regions before noise removal (d) Foreground regions after noise removal

After experimenting with different combinations of these operations, we have come up with the following sequence: two-levels of dilation followed by three-levels of erosion and finally one-level of dilation. The first dilation operation removes the holes (NBN) in foreground objects that are detected as background and expands the regions' boundaries. In the next step, three-levels of erosion removes the extra pixels on the region boundaries generated by the previous step and removes isolated noisy regions (NFN). The last step, one level of dilation, is used to compensate the one-level extra effect of erosion. Figure 3.5 shows sample foreground regions before and after noise removal together with original image. Note that the resolution of actual image (320 x 240) is different than the one used for foreground detection (160 x 120).

Removal of shadow regions and detecting and adapting to sudden illumination changes require more advanced methods which are explained in the next section.

3.1.2.1 Shadow and Sudden Illumination Change Detection

Most of the foreground detection algorithms are susceptible to both shadows and sudden illumination changes which cause inaccurate foreground object segmentation. Since later processing steps like object classification and tracking depend on the correctness of object segmentation, it is very important to cope with shadow and sudden illumination changes in smart surveillance systems.

In our system we used a shadow detection scheme which is inspired from the work presented in [21]. We make use of the fact that for pixels in shadow regions the RGB color vectors are in the same direction with the RGB color vectors of the corresponding background pixels with a little amount of deviation and the shadow pixel's brightness value is less than the corresponding background pixel's brightness. In order to define this formally, let I_x represent the RGB color of a current image pixel at position x, and B_x represent the RGB color of the corresponding background pixel. Furthermore, let $\hat{I}_x$ represent the vector that start at the origin O (0, 0, 0) in RGB color space and end at point I_x, let $\hat{B}_x$ be the vector for corresponding background pixel B_x and let d_x represent the dot product $(\cdot)$ between $\hat{I}_x$ and $\hat{B}_x$. Figure 3.6 show these points and vectors in RGB space. Our shadow detection scheme classifies a pixel that is part of the detected foreground as shadow if it satisfies:

$$\left(d_x = \frac{\hat{I}_x}{\left\|\hat{I}_x\right\|} \cdot \frac{\hat{B}_x}{\left\|\hat{B}_x\right\|} \right) < \tau \tag{3.14}$$

and

$$\left\|\hat{I}_x\right\| < \left\|\hat{B}_x\right\| \tag{3.15}$$

where τ is a pre-defined threshold which close to one. Dot product is used to test whether $\hat{I}_x$ and $\hat{B}_x$ have the same direction or not. If the dot product (d_x) of normalized $\hat{I}_x$ and $\hat{B}_x$ is close to one, this implies that they are almost in the

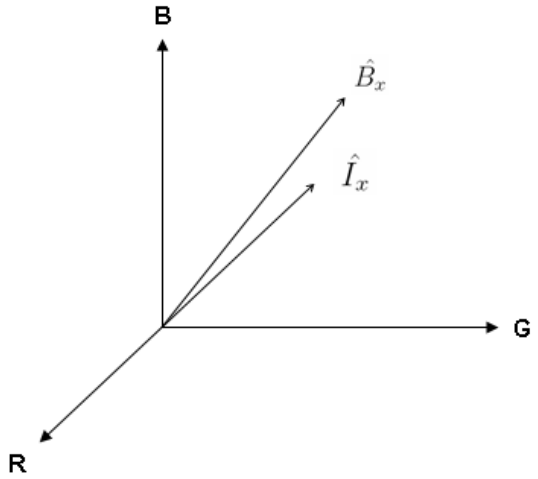
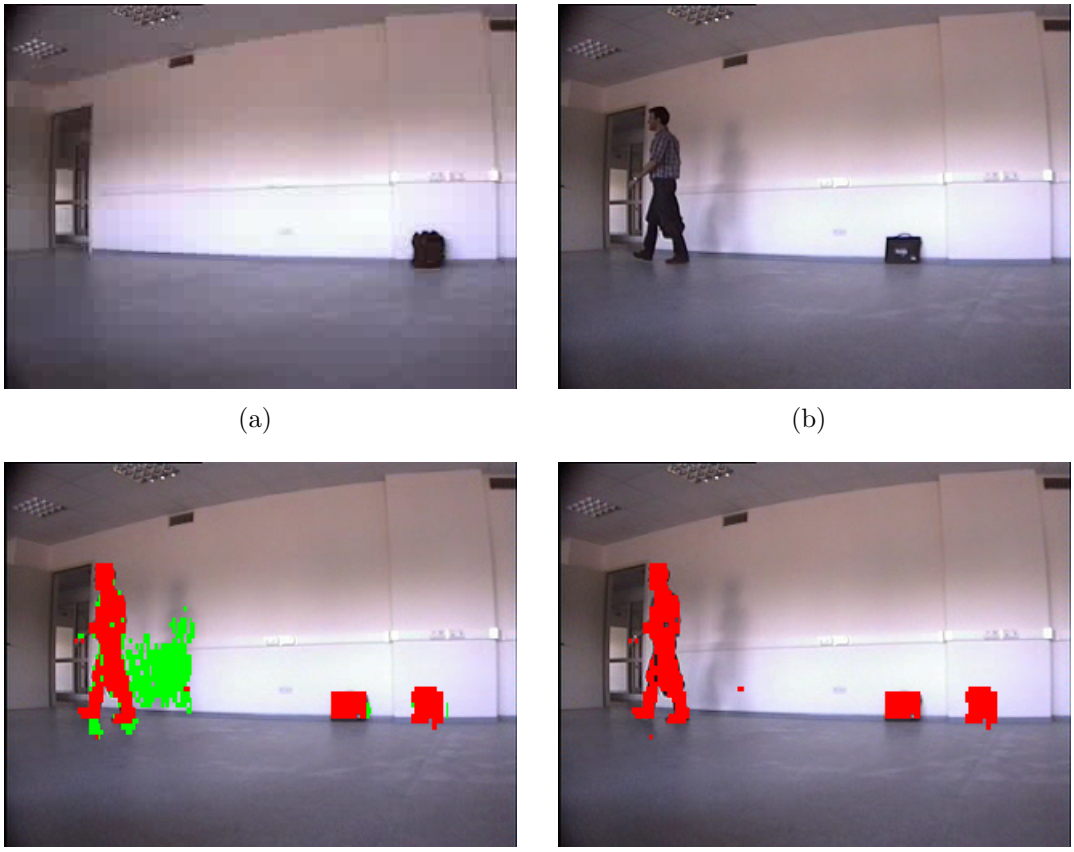


Figure 3.6: RGB vectors of current image pixel $\hat{I}_x$ and corresponding background pixel $\hat{B}_x$.

same direction with a little amount of deviation. The second check is performed to ensure that the brightness value of I_x is less than B_x. Figure 3.7 shows sample foreground regions with shadows before and after shadow removal.

Besides shadow removal, sudden illumination change detection is also a requirement that needs to be met by a smart surveillance system to continue detecting and analyzing object behavior correctly. A global change may for instance occur due to sun being covered/uncovered by clouds in outdoor environments or due to turning lights on in an indoor environment. Both of these changes make a sudden brightness change in the scene which even adaptive background models cannot handle. Figure 3.8 shows sample frames before and after a sudden light change. Our method of sudden light change detection makes use of the same observation used in [37, 55], which is the fact that the sudden global light change causes the background models to classify a big proportion ($\geq 50\%$) of the pixels in the scene as foreground. However, in some situations, where ordinary objects move very close to the camera, this assumption is too simplistic and fails. Thus, for the aim of distinguishing a global light change from large object motion, we make another check by exploiting the fact that in case of a global light change, the topology of the object edges in the scene does not change too much and the



(a) (b)

(c) (d)

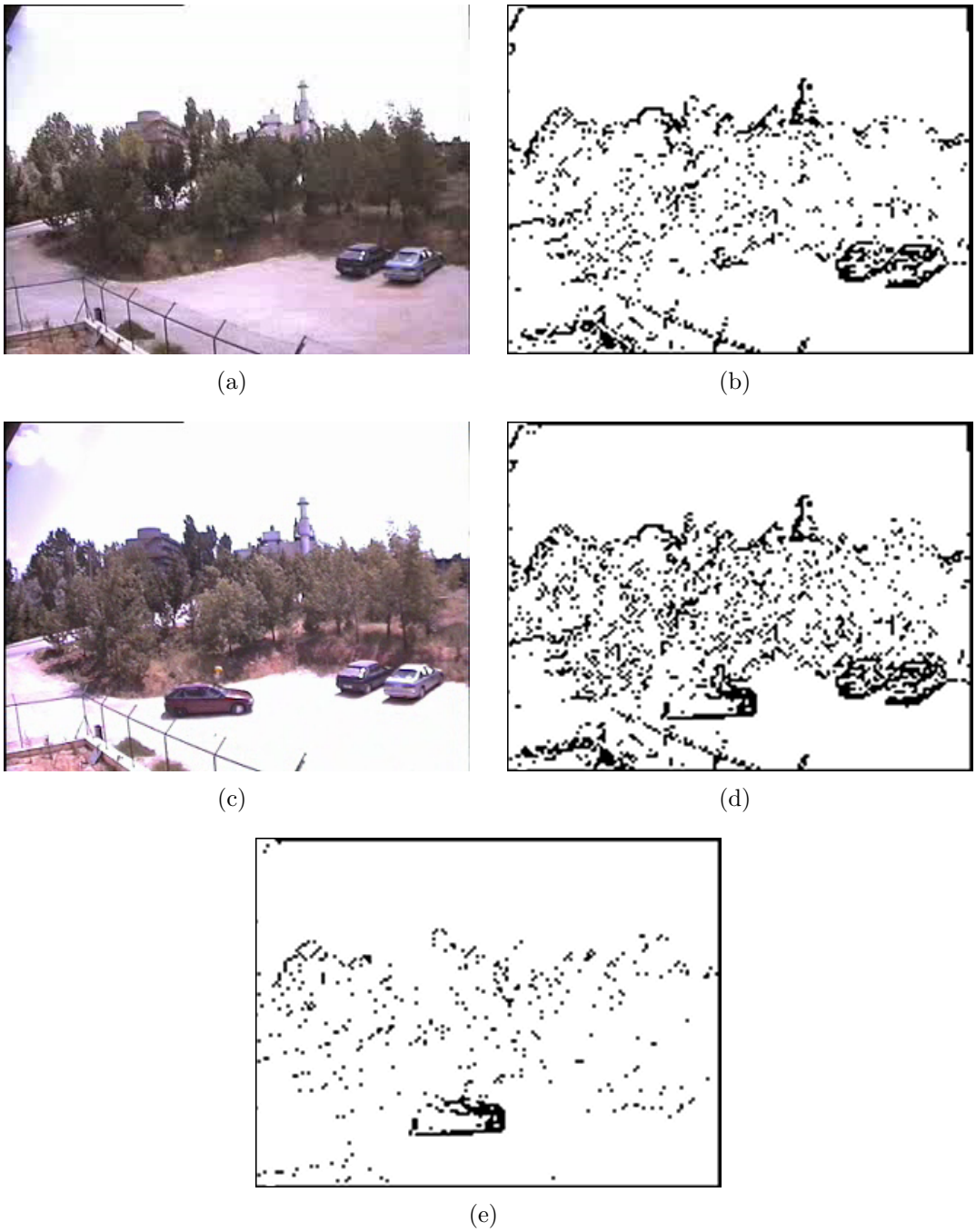
Figure 3.7: Shadow removal sample. (a) Estimated background (b) Current image (c) Detected foreground pixels (shown as red) and shadow pixels (shown as green) (d) Foreground pixels after shadow pixels are removed

(a) (b)

Figure 3.8: Sudden light change sample. (a) The scene before sudden light change (b) The same scene after sudden light change

boundaries of the detected foreground regions do not correspond to actual edges in the scene whereas in case of large object motion the boundaries of the detected foreground regions correspond to the actual edges in the image.

In order to check whether the boundaries of the detected regions correspond to actual edges in the current image, we utilize the gradients of current image and the background image. The gradients are found by taking the brightness difference between consecutive pixels in the images in both horizontal and vertical directions. After the gradients are found both for background and current image, a threshold is applied and the output is converted to binary (where a one represents an edge). Then, the difference image of background and current image gradients is calculated to find only the edges that correspond to moving regions. Figure 3.9 shows sample gradient images for background and current images. Finally, the detected foreground region is eroded from outside towards inside till hitting an edge pixel in the gradient difference image. If the resulting foreground region is very small compared to the original, then this is an indication of a global light change, hence the background model is re-initiated with current and following few images. Wavelet images can also be used instead of gradients to distinguish a sudden global light change.



(a) (b) (c) (d) (e)

Figure 3.9: Detecting true light change. (a) Estimated reference background (b) Background's gradient (c) Current image (d) Current image's gradient (e) The gradient difference

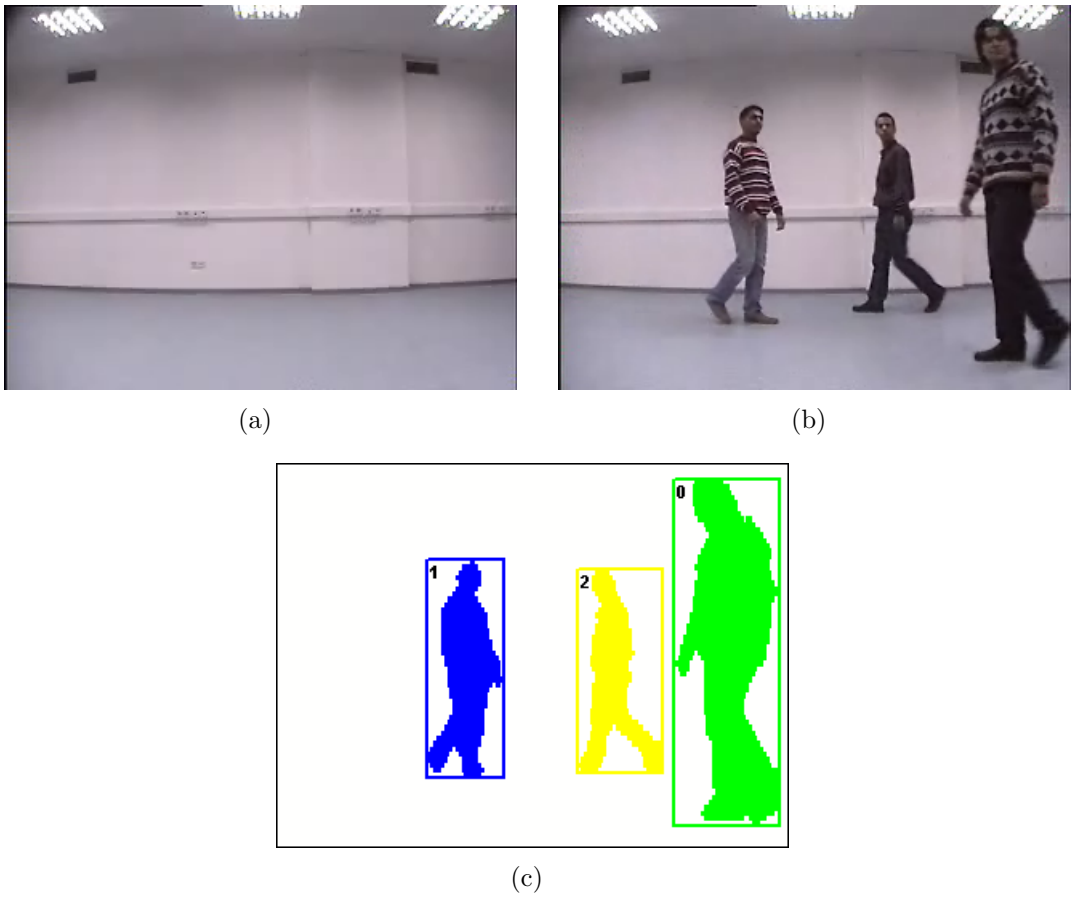


Figure 3.10: Connected component labeling sample. (a) Estimated background (b) Current image (c) Filtered foreground pixels and connected and labeled regions with bounding boxes

3.1.3 Detecting Connected Regions

After detecting foreground regions and applying post-processing operations to remove noise and shadow regions, the filtered foreground pixels are grouped into connected regions (blobs) and labeled by using a two-level connected component labeling algorithm presented in [19]. After finding individual blobs that correspond to objects, the bounding boxes of these regions are calculated. Figure 3.10 shows sample foreground regions before and after region connecting, labeling and boxing.

3.1.4 Region Level Post-Processing

Even after removing pixel-level noise, some artificial small regions remain due to inaccurate object segmentation. In order to eliminate this type of regions, the average region size (γ) in terms of pixels is calculated for each frame and regions that have smaller sizes than a fraction (α) of the average region size ($Size(region) < \alpha * \gamma$) are deleted from the foreground pixel map.

Also, due to segmentation errors, some parts of the objects are found as disconnected from the main body. In order to correct this defect, the bounding boxes of regions that are close to each other are merged together and the region labels are adjusted.

3.1.5 Extracting Object Features

Once we have segmented regions we extract features of the corresponding objects from the current image. These features are size (S), center of mass (C_m), color histogram (H_c) and silhouette contour of the object's blob. Calculating the size of the object is trivial and we just count the number of foreground pixels that are contained in the bounding box of the object.

In order to calculate the center of mass point, $C_m = (x_{C_m}, y_{C_m})$, of an object O, we use the following equation[42]:

$$x_{C_m} = \frac{\sum_i^n x_i}{n}, \;\; y_{C_m} = \frac{\sum_i^n y_i}{n} \tag{3.16}$$

where n is the number of pixels in O.

The color histogram, H_c is calculated over monochrome intensity values of object pixels in current image. In order to reduce computational complexity of operations that use H_c, the color values are quantized. Let N be the number of bins in the histogram, then every bin covers $\frac{255}{N}$ color values.

The color histogram is calculated by iterating over pixels of O and incrementing the stored value of the corresponding color bin in the histogram, H_c. So for an object O the color histogram is updated as follows:

$$H_c[\frac{c_i}{N}] = H_c[\frac{c_i}{N}] + 1, \quad \forall \ c_i \in O \tag{3.17}$$

where c_i represents the color value of i^{th} pixel. In the next step the color histogram is normalized to enable appropriate comparison with other histograms in later steps. The normalized histogram $\hat{H}_c$ is calculated as follows:

$$\hat{H}_c[i] = \frac{H_c[i]}{\sum_i^N H_c[i]} \tag{3.18}$$

After experimenting with histogram based object tracking in later steps, we realized that calculating the color histogram by all of the pixels of an object may fail the algorithm to resolve identities correctly. The reason is that, for instance, if two people have the same colored clothes, e.g. one has a white shirt and black trousers and the other one has a black shirt and white trousers, the histogram calculated over whole bodies will include almost the same amount of white and black for both objects. Hence, we decided to calculate two histograms for an object, one for the upper body (H_c^u) and one for the lower body (H_c^l). Silhoutte contour extraction will be discussed later in object classification in Section 4.1.1.

3.2 Object Tracking

The aim of object tracking is to establish a correspondence between objects or object parts in consecutive frames and to extract temporal information about objects such as trajectory, posture, speed and direction. Tracking detected objects frame by frame in video is a significant and difficult task. It is a crucial part of smart surveillance systems since without object tracking, the system could not extract cohesive temporal information about objects and higher level behavior analysis steps would not be possible. On the other hand, inaccurate foreground object segmentation due to shadows, reflectance and occlusions makes tracking a difficult research problem.

We used an object level tracking algorithm in our system. That is, we do not track object parts, such as limbs of a human, but we track objects as a whole from frame to frame. The information extracted by this level of tracking is adequate for most of the smart surveillance applications.

The tracking method we have developed is inspired by the study presented in [2]. Our approach makes use of the object features such as size, center of mass, bounding box and color histogram which are extracted in previous steps to establish a matching between objects in consecutive frames. Furthermore, our tracking algorithm detects object occlusion and distinguishes object identities after the split of occluded objects. By analyzing the object trajectory information, our tracking system is able to detect left and removed objects as well. The system diagram for our tracking method is shown in Figure 3.11.

3.2.1 Correspondence-based Object Matching

The activity diagram of our correspondence-based object matching algorithm is shown in Figure 3.12. The first step in our object tracking algorithm is matching the objects (O_p's) in previous image (I_{n-1}) to the new objects (O_i's) detected in current image (I_n).

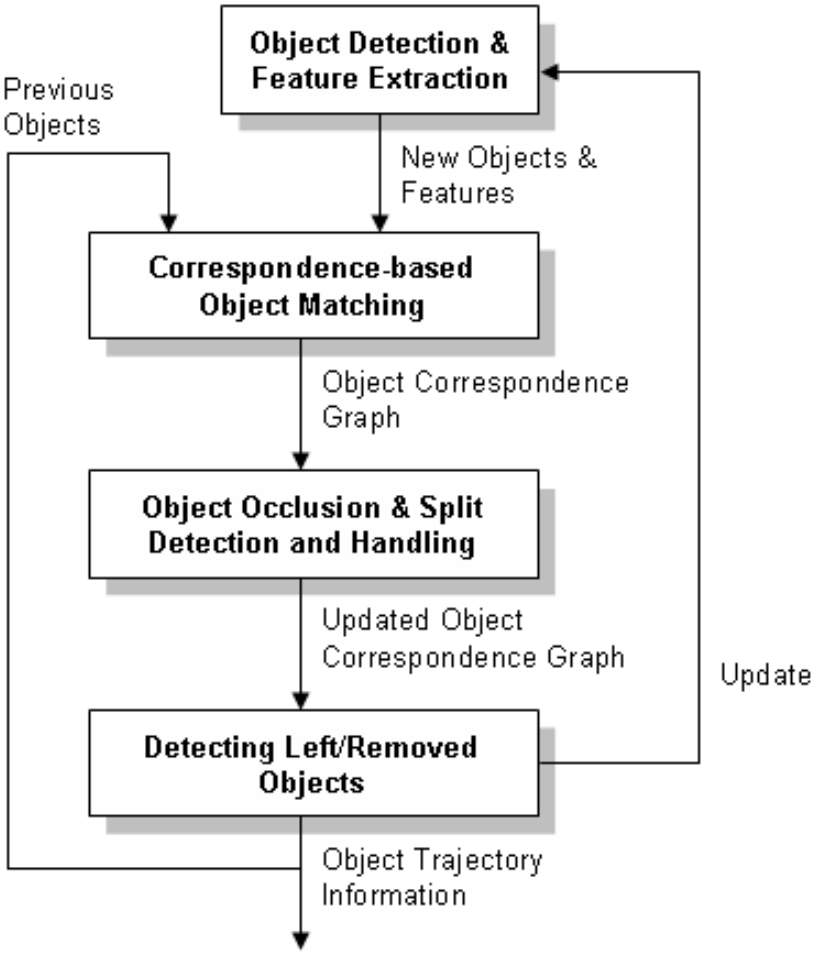


Figure 3.11: The object tracking system diagram.

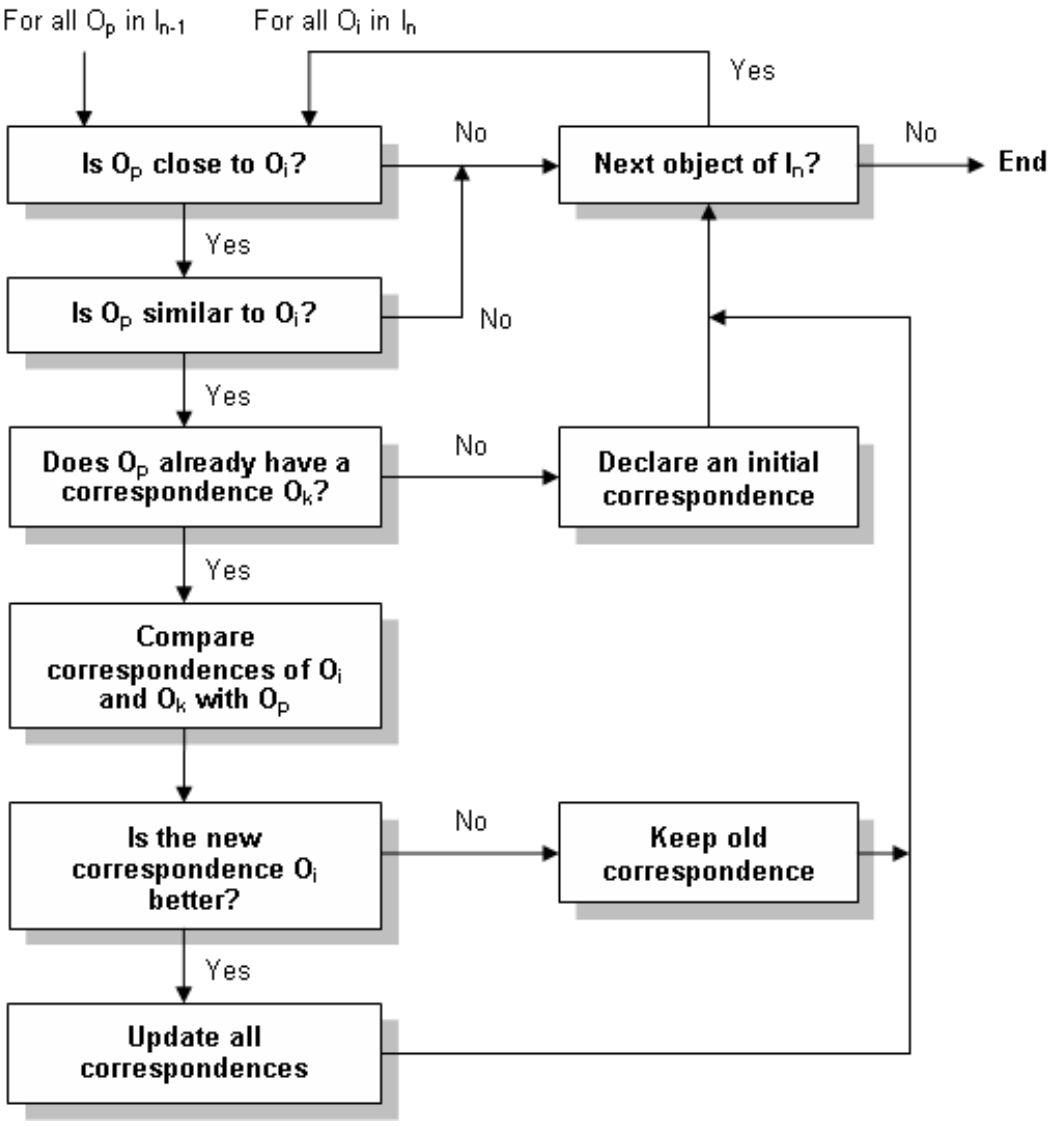


Figure 3.12: The correspondence-based object matching method.

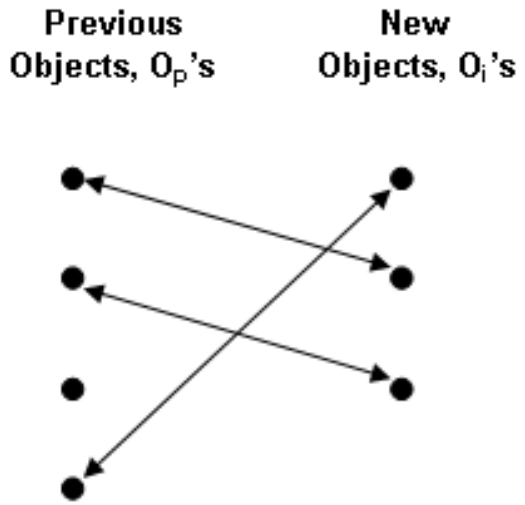


Figure 3.13: Sample object matching graph.

We store the matching of objects in a bi-partite graph $G(m, n)$. In this graph, vertices represent the objects (one vertex partition represents previous objects, O_p's and the other partition represents new objects, O_i's) and edges represent a match between two objects. In $G(m, n)$, m is the size of the partition for previous objects, and n is the size of the partition for the new objects. A simple matching graph is shown in Figure 3.13. In order to perform object matching, we iterate over the list of previous objects and new objects to evaluate their correspondences. For each previous object, O_p we iterate over new objects and first check whether a new object O_i in the new objects list is close to O_p or not. The criterion for closeness is defined as the distance between the center of mass points of these two objects (O_p and O_i) being smaller than a pre-defined constant. This check is inspired by the fact that the displacement of an object between consecutive images should be small. In other words, two objects with center of mass points c_p and c_i are close to each other if the following is satisfied:

$$Dist(c_p, c_i) < \tau \tag{3.19}$$

where $Dist()$ function is defined as the Euclidean distance between two points, which is:

$$Dist(c_p, c_i) = \sqrt{(x_{c_p} - x_{c_i})^2 + (y_{c_p} - y_{c_i})^2} \tag{3.20}$$

Since every two objects that are close to each other within a threshold are not necessarily a successful match, in the next step we check the similarity of these two objects to improve correct matching. The criterion for similarity comparison is the size ratio of the objects. Again, this check is motivated by the fact that objects do not grow or shrink too much between consecutive frames. Thus, two objects are classified as similar if they satisfy the following:

$$\frac{s_p}{s_i} < \mu \quad or \quad \frac{s_i}{s_p} < \mu \tag{3.21}$$

where s_i is the size of object O_i and μ is a pre-defined threshold. Checking the objects for size is especially useful if an object in the previous frame splits into a large and a very small region due to inaccurate segmentation. This check eliminates the chance of matching a big region to a small region.

If we only performed the above two steps, we would come up with cases where a previous object is matched to more than one new object. Hence, after the second step we check further whether the object O_p has already a match/correspondence or not. If the object O_p does not have prior-correspondence, we connect the corresponding vertices in the bi-partite graph $G(m, n)$ and continue with next new object O_i, but if O_p has a prior-correspondence O_k, we perform additional steps to resolve the correspondence conflict.

In resolving a matching conflict we compare the correspondences of objects O_i and O_k to O_p. In other words, by comparing the correspondence of O_i and O_p with the correspondence of O_k and O_p, we try to decide which one of O_i or O_k is the correct match to object O_p. The correspondences are compared by using the distance between the center of mass point of O_p and O_i or O_k. Let d_{pi} be the distance between center of mass points of O_p and O_i, and let d_{pk} be the distance between center of mass points of O_p and O_k. The correspondence is resolved in favor of O_k if $d_{pk} < d_{pi}$, otherwise resolution is in favor of O_i. We might have used stronger criteria in correspondence matching, such as color histogram comparison; however in our experiments using distance for resolution performed well.

Another conflict arises for the case if O_i has a prior-correspondence established in previous iteration over the previous objects list. For instance, O_{p-1} might have been matched with O_i, and in the next iteration for O_p, it is possible that the first two checks will be satisfied between O_p and O_i and O_i would be assigned to O_p. However, we know that O_i has already a correspondence. Hence, this causes similar correspondence conflict and we resolve this again by using the distance-based scheme explained in previous paragraph.

In establishing a matching between previous objects and new objects five different match cases can occur. The details of these cases and their handling by our tracking method are explained below.

1. *One-to-one*: This is the case where previous object O_p is matched with a single new object O_i. The features of O_p is updated with incoming information from O_i.

2. *One-to-many*: This is the case where previous object O_p is matched with more than one new object. This conflicting case is resolved by distance-based correspondence comparison and it reduces to case 1.

3. *One-to-none*: This is the case where previous object O_p is not matched to any new object. This case occurs if an object disappears from the scene or if the object is occluded by other objects. In case of an occlusion, the object is preserved until the detection of the corresponding occlusion split. Otherwise, this object is deleted from the previous objects list.

4. *None-to-one*: This is the case where new object O_i is not matched to any of the existing objects. This case occurs if a new object enters into the scene or occluded objects split. In case of an occlusion split, the corresponding object is found by the occlusion handling procedure which will be explained in the next sections. If this is due to a new object, the object O_i is added to the tracked objects list.

5. *Many-to-one*: This is the case where new object O_i is matched with more than one previous object. This conflicting case is resolved by distance-based correspondence comparison and it reduces to case 1.

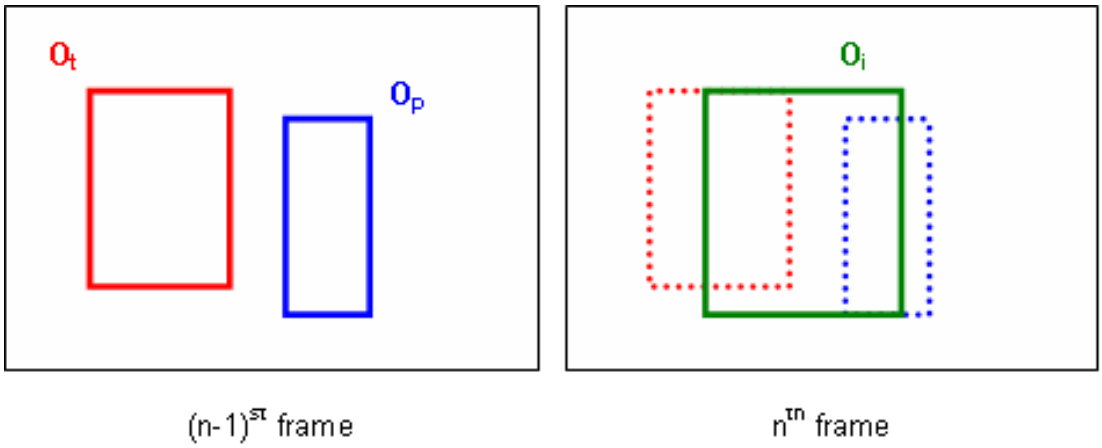


Figure 3.14: Occlusion detection sample case.

3.2.2 Occlusion Handling

Most of the object detection methods are not able to detect occlusions among objects. Therefore, special techniques are required to continue tracking objects even in occlusion cases. Our tracking system uses simple heuristics to detect object occlusions and occlusion group splits and to distinguish object identities (which object is which?) after occlusions. Details of these three steps are described in the following sections.

3.2.2.1 Detecting Object Merge

We make use of a simple assumption in detecting occlusions. When an object O_p is found to disappear by the initial object matching algorithm (case 2 in Section 3.2.1), we check whether there is a new object O_i whose bounding box is overlapping with that of O_p and which is matched to a previous object O_t. In such a case, it is highly possible that O_p and O_t are occluded with each other and formed a new object O_i. Figure 3.14 shows a sample case. After detecting such a case, we do not delete object O_p from the previous objects list but mark it as occluded. We create an occlusion group from the objects that are occluded

with each other and assign a new occlusion group ID to these objects. For the case if one of the occluding objects has already an occlusion group ID, we merge these different occlusion groups into one. We also, store the pre-occlusion color histograms of objects in order to use in identification process after a split.

3.2.2.2 Detecting Object Split

Detecting an occluded object split utilizes a similar heuristic as occlusion detection. When an object O_i is found to enter to the scene by the initial object matching algorithm (case 4 in Section 3.2.1), we check whether there was a previous object O_t whose bounding box is overlapping with that of O_i and who has a valid occlusion group ID and who is matched to another new object O_k. In such a case, this might be considered as an object split. We check the previous object list for objects that have the same occlusion group ID as O_t. Assume that we found O_p to have the same occlusion group ID with O_p which means that O_p and O_t were occluded by each other previously. We then have two tracking objects $TO = \{O_p, O_t\}$ and two new objects $NO = \{O_i, O_k\}$. Now we need to identify which object in TO corresponds to which object in NO.

3.2.2.3 Histogram-based Correspondence Matching

In order to match objects in TO to the ones in NO, we make use of the stored pre-occlusion color histograms of tracking objects and color histograms of new objects. We perform a matching scheme similar to the initial object matching method explained before. However, since we cannot match objects based on their position, using distance is not feasible in this case. Hence, to compare correspondences of objects, we use color histogram distance.

The distance d_{ab} between two normalized color histograms H_a and H_b with N bins are calculated by using the L_1 metric as follows:

$$d_{ab} = \sum_{i}^{N} |H_a[i] - H_b[i]| \tag{3.22}$$

Since we keep two histograms per object, one for upper body and one for lower body, we calculate the total distance by summing up the distances between corresponding color histograms of objects. That is:

$$d_{total} = d_{upper\ histogram} + d_{lower\ histogram} \tag{3.23}$$

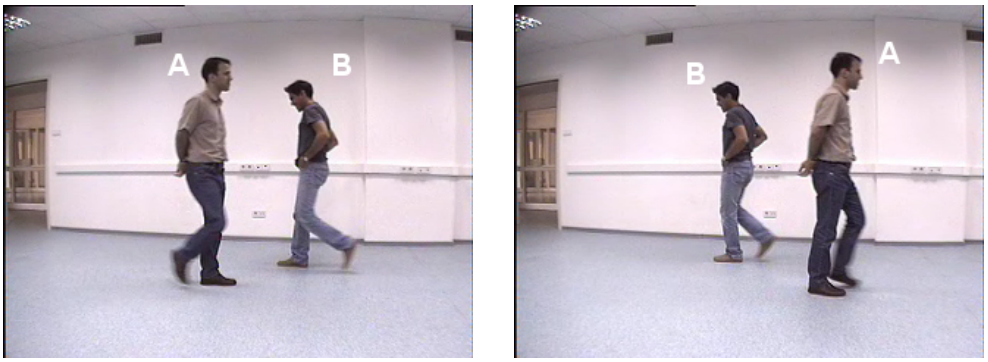
Figure 3.15 shows sample objects and histograms before and after an occlusion and their distance table. The objects with minimum color histogram distance are matched together. Conflicts are again resolved by using the color histogram distance.

3.2.3 Detecting Left and Removed Objects

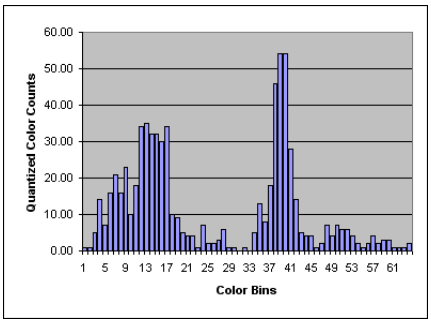
The ability of detecting left and removed objects in a scene is unconditionally vital in some visual surveillance applications. Detecting left objects such as unattended luggage in airports or a car parked in front of a security sensitive building is important since these activities might be performed by terrorists to harm people. On the other hand, protecting objects against removal without permission has important applications such as in surveillance of museums, art galleries or even department stores to prevent theft. Due to these critical applications, left/removed object is important part of a surveillance system.

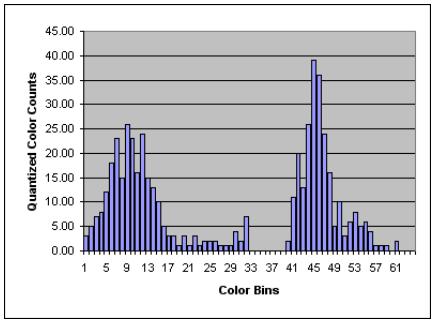
Our system is able to detect and distinguish left and removed objects in video imagery. To accomplish this, we use our adaptive background subtraction scheme, object tracking method and a heuristic to distinguish left objects from removed ones. The three steps in detecting left or removed objects is as follows:

1. Detecting a change between the current image and the reference background image by using the adaptive background subtraction scheme.
2. Deciding that the detected region corresponds to a left or removed object by using object tracking method.
3. Distinguishing the left objects from removed objects by using the statistical color property of the detected and its surrounding regions.

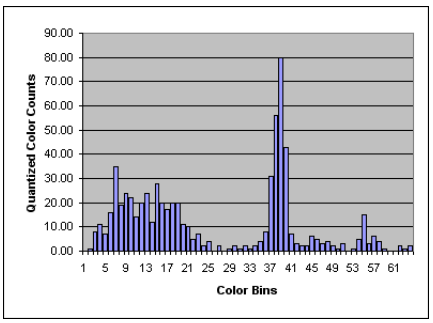


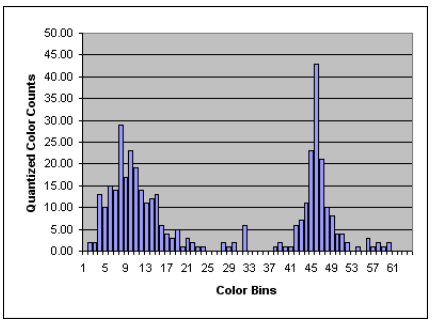
(a) (b)





(c) (d)





(e) (f)

	Object A After	Object B After
Object A Before	0.283156	0.467758
Object B Before	0.358641	0.241878

(g)

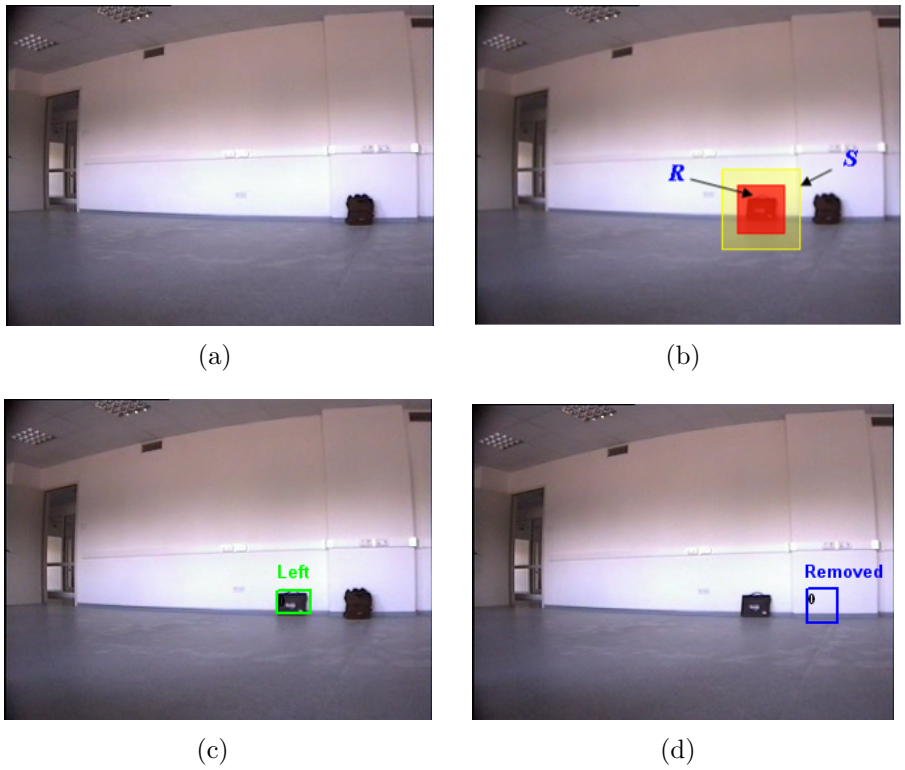
Figure 3.15: Object identification after occlusion. (a) Image before occlusion (b) Image after occlusion (c) Color histogram of object A before occlusion (d) Color histogram of object B before occlusion (e) Color histogram of object A after occlusion (f) Color histogram of object B after occlusion (g) Normalized color histogram distance table of objects A and B

Unlike some other algorithms, for instance temporal differencing, our adaptive background subtraction algorithm is able to detect objects being left or removed to/from the background scene for a long period of time. With the help our tracking method, we detect that the object is stationary by using its trajectory information. If the recent part of the trajectory information states that the object has not moved for a long time (e.g. alarm period), we decide that the corresponding region is stationary and is possibly a candidate of being a left or removed object.

In order to distinguish the type of the object (left or removed) we use the statistical properties of the color values in and around the detected region. Let R represent the region corresponding to a long term change in the background; S represent the surrounding region around R and let A_X represent the average color intensity value in a region X. Our heuristic we developed by experimenting several left/removed object video states that if the values of A_R and A_S are close to each other, then this indicates that the detected object region and its surrounding region has almost the same color and therefore the region corresponds to a removed object. If on the other hand A_R and A_S are not close to each other this indicates that the region corresponds to a left object. We decide whether A_R is close to A_S or not as follows:

$$\begin{aligned} \tau \leq \tfrac{A_R}{A_S} \leq 1, \quad if A_R \leq A_S \\ \tau \leq \tfrac{A_S}{A_R} \leq 1, \quad if A_S \leq A_R \end{aligned} \tag{3.24}$$

where τ is a pre-defined constant (≈ 0.85). Figure 3.16 depicts a drawing to show the regions A_R and A_S and two sample video images which show left and removed object cases.



(a) (b)

(c) (d)

Figure 3.16: Distinguishing left and removed objects. (a) Scene background (b) Regions R and S (c) Left object sample (d) Removed object sample

Chapter 4

Object Classification

The ultimate aim of different smart visual surveillance applications is to extract semantics from video to be used in higher level activity analysis tasks. Categorizing the type of a detected video object is a crucial step in achieving this goal. With the help of object type information, more specific and accurate methods can be developed to recognize higher level actions of video objects. Hence, we developed a novel video object classification method based on object shape similarity as part of our visual surveillance system.

Typical video scenes may contain a variety of objects such as people, vehicles, animals, natural phenomenon (e.g. rain, snow), plants and clutter. However, main target of interest in surveillance applications are generally humans and vehicles. Also, real time nature and operating environments of visual surveillance applications require a classification scheme which is computationally inexpensive, reasonably effective on small targets and invariant to lighting conditions [12]. We have satisfied most of these requirements by implementing a classification scheme which is able to categorize detected video objects into pre-defined groups of human, human group and vehicle by using image-based object features.

4.1 Silhouette Template Based Classification

The classification metric used in our method measures object similarity based on the comparison of silhouettes of the detected object regions extracted from the foreground pixel map with pre-labeled (manually classified) template object silhouettes stored in a database. The whole process of object classification method consists of two steps:

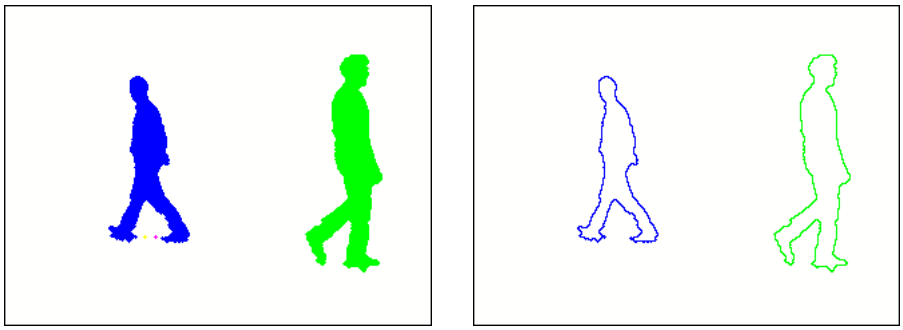
- *Offline* step: Creating a template database of sample object silhouettes by manually labeling object types.
- *Online* step: Extracting the silhouette of each detected object in each frame and recognizing its type by comparing its silhouette based feature with the ones in the template database in real time during surveillance. After the comparison of the object with the ones in the database, a template shape with minimum distance is found. The type of this object is assigned to the type of the object which we wanted to classify. In this step the result of object tracking step is utilized to attain temporal consistency of classification results.

4.1.1 Object Silhouette Extraction

Both in offline and online steps of the classification algorithm, the silhouettes of the detected object regions are extracted from the foreground pixel map by using a contour tracing algorithm presented in [19]. Figure 4.1 shows sample detected foreground object regions and the extracted silhouettes.

4.2 Silhouette Template Database

The template silhouette database is created offline by extracting several object contours from different scenes. Since the classification scheme makes use of object



(a) (b)

Figure 4.1: Sample detected foreground object regions and extracted silhouettes.

similarity, the shapes of the objects in the database should be representative poses of different object types. Considering human type, we add human shapes in different poses to the template database in order to increase the chance of a query object of type human to be categorized correctly. For instance, if we all have human shapes in erect positions; we may miss categorizing a human which is sitting on a chair. Or if we have silhouettes of cars all are viewed horizontally from the camera, we may miss to classify vehicles moving vertically with respect to the camera view. Figure 4.2 shows a small template database of size 24 having different poses for human, human group and vehicles.

In classification step, our method does not use silhouettes in raw format, but rather compares converted silhouette distance signals. Hence, in the template database we store only the distance signal of the silhouette and the corresponding type information for both computational and storage efficiency.

Let $S = \{p_1, p_2, \ldots, p_n\}$ be the silhouette of an object O consisting of n points ordered from top center point of the detected region in clockwise direction and c_m be the center of mass point of O. The distance signal $DS = \{d_1, d_2, \ldots, d_n\}$ is generated by calculating the distance between c_m and each p_i starting from 1 through n as follows:

$$d_i = Dist(c_m, p_i), \quad \forall \ i \in [1 \ldots n] \tag{4.1}$$

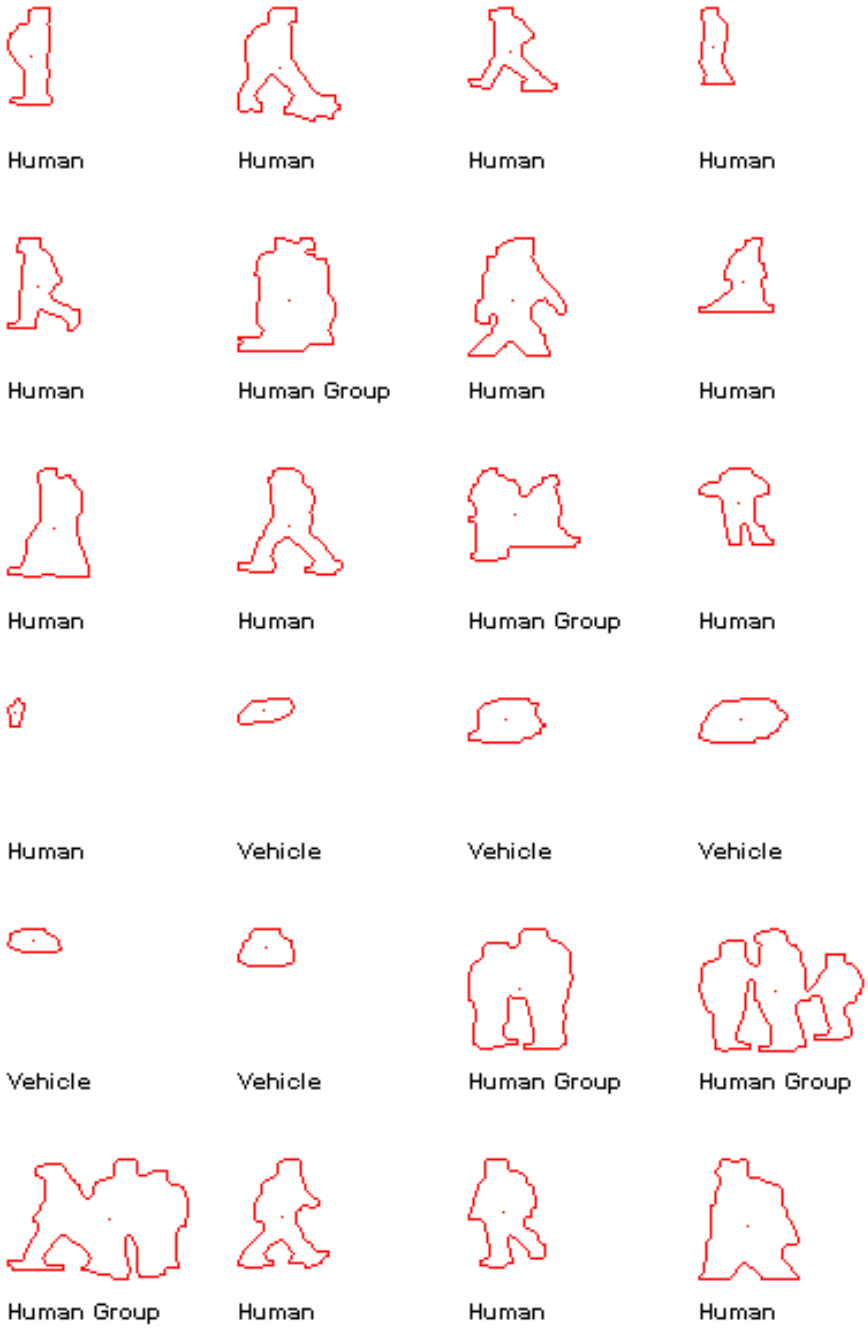


Figure 4.2: Sample silhouette template database with labels.

where the $Dist$ function is the Euclidian distance between two points a and b:

$$Dist(a,b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{4.2}$$

Different objects have different shapes in video and therefore have silhouettes of varying sizes. Even the same object has altering contour size from frame to frame. In order to compare signals corresponding to different sized objects accurately and to make the comparison metric scale-invariant we fix the size of the distance signal. Let N be the size of a distance signal DS and let C be the constant for fixed signal length. The fix-sized distance signal $\widehat{DS}$ is then calculated by sub-sampling or super-sampling the original signal DS as follows:

$$\widehat{DS}[i] = DS[i * \frac{N}{C}], \quad \forall \ i \in [1 \dots C] \tag{4.3}$$

In the next step, the scaled distance signal $\widehat{DS}$ is normalized to have integral unit area. The normalized distance signal $\overline{DS}$ is calculated with the following equation:

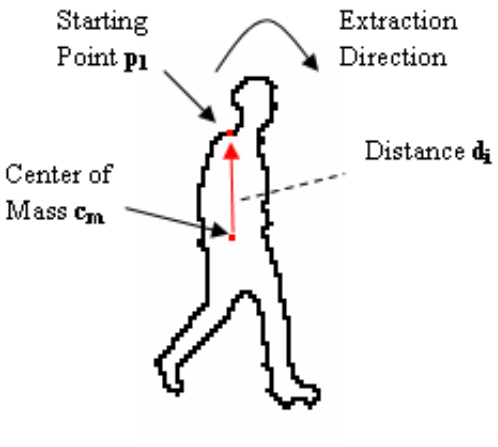
$$\overline{DS}[i] = \frac{\widehat{DS}[i]}{\sum_1^n \widehat{DS}[i]} \tag{4.4}$$

Figure 4.3 shows a sample silhouette and its original and scaled distance signals.

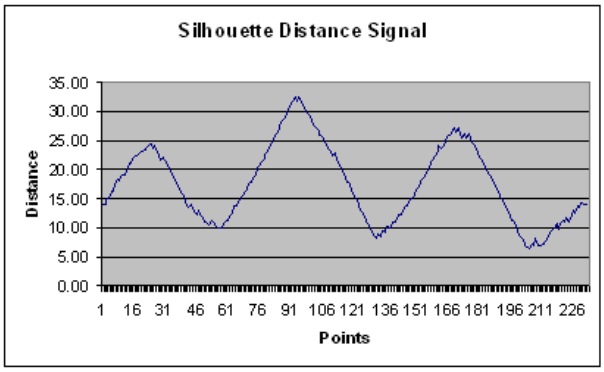
4.3 The Classification Metric

Our object classification metric is based on the similarity of object shapes. There are numerous methods in the literature for comparing shapes [43, 7, 42, 3, 22]. The reader is especially referred to the surveys presented in [47, 31] for good discussions on different techniques.

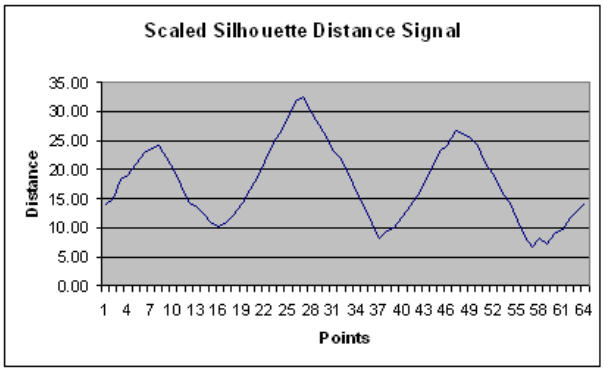
The important requirements of a shape comparison metric are scale, translation and rotation invariance. Our method satisfies all three of these properties.



(a)



(b)



(c)

Figure 4.3: Sample object silhouette and its corresponding original and scaled distance signals. (a) Object silhouette (b) Distance signal (c) Scaled distance signal

1. *Scale invariance*: Since we use a fixed length for the distance signals of object shapes, the normalized-and-scaled distance signal will almost be the same for two different representations (in different scales) of the same pose of an object.

2. *Translation invariance*: The distance signal is independent of the geometric position of the object shape since the distance signal is calculated with respect to the center of mass of the object shape. Due to the fact that the translation of the object shape will not change the relative position of the center of mass point's position with respect to the object, the comparison metric will not be affected by translation.

3. *Rotation invariance*: We do not use the rotation invariance property of our classification metric since we want to distinguish even the different poses of a single object for later steps in the surveillance system. However, by choosing a different starting point p_s on the silhouette of the object in contour tracing step, we could calculate distance signals of the object for different rotational transformations for each starting point p_s.

Our classification metric compares the similarity between the shapes of two objects, A and B, by finding the distance between their corresponding distance signals, $\overline{DS}_A$ and $\overline{DS}_B$. The distance between two scaled and normalized distance signals, $\overline{DS}_A$ and $\overline{DS}_B$ is calculated as follows:

$$Dist_{AB} = \sum_{i=1}^{n} \left| \overline{DS}_A[i] - \overline{DS}_B[i] \right| \tag{4.5}$$

In order to find the type T_O of an object O, we compare its distance signal $\overline{DS}_O$ with all of the objects' distance signals in the template database. The type T_P of the template object P is assigned as the type of the query object O, $T_O = T_P$ where P satisfies the following:

$$Dist_{OP} \leq Dist_{OI}, \quad \forall \text{ object } I \text{ in the template database} \tag{4.6}$$

Figure 4.4 shows the silhouettes, silhouette signals and signal distances of a sample query object and template database objects for type classification.

4.4 Temporal Consistency

The performance of the object classification method is dependent on the quality of the output of the object segmentation step. Due to environmental factors, such as objects being occluded by stationary foreground objects (e.g. a fence or a pole in front of the camera) or due to the fact that only a part of the object is entered into the scene, the shape of the detected region does not reflect an object's true silhouette. In such cases, the classification algorithm fails to label the type of the object correctly. For instance, the part of a vehicle entering into the scene may look like a human, or a partially occluded human may look like a human group. Therefore, we use a multi-hypothesis scheme [29] to increase the accuracy of our classification method.

In this process, a type histogram H_T is initialized and maintained for an object O detected in the scene. The size of this histogram is equal to the number of different object types (e.g. three in our system representing human (H), human group (HG) and vehicle (V)) and each bin i of this histogram keeps the number of times the object O is found of type T_i (one of H, HG, V). Figure 4.5 shows a sample object and its type histogram for three different frames.

With the help of this multiple hypothesis scheme, possible types of an object can be accumulated over a pre-defined period of time and the true decision of its type can be made more accurately by selecting the type of the bin with biggest value as the type of the object.

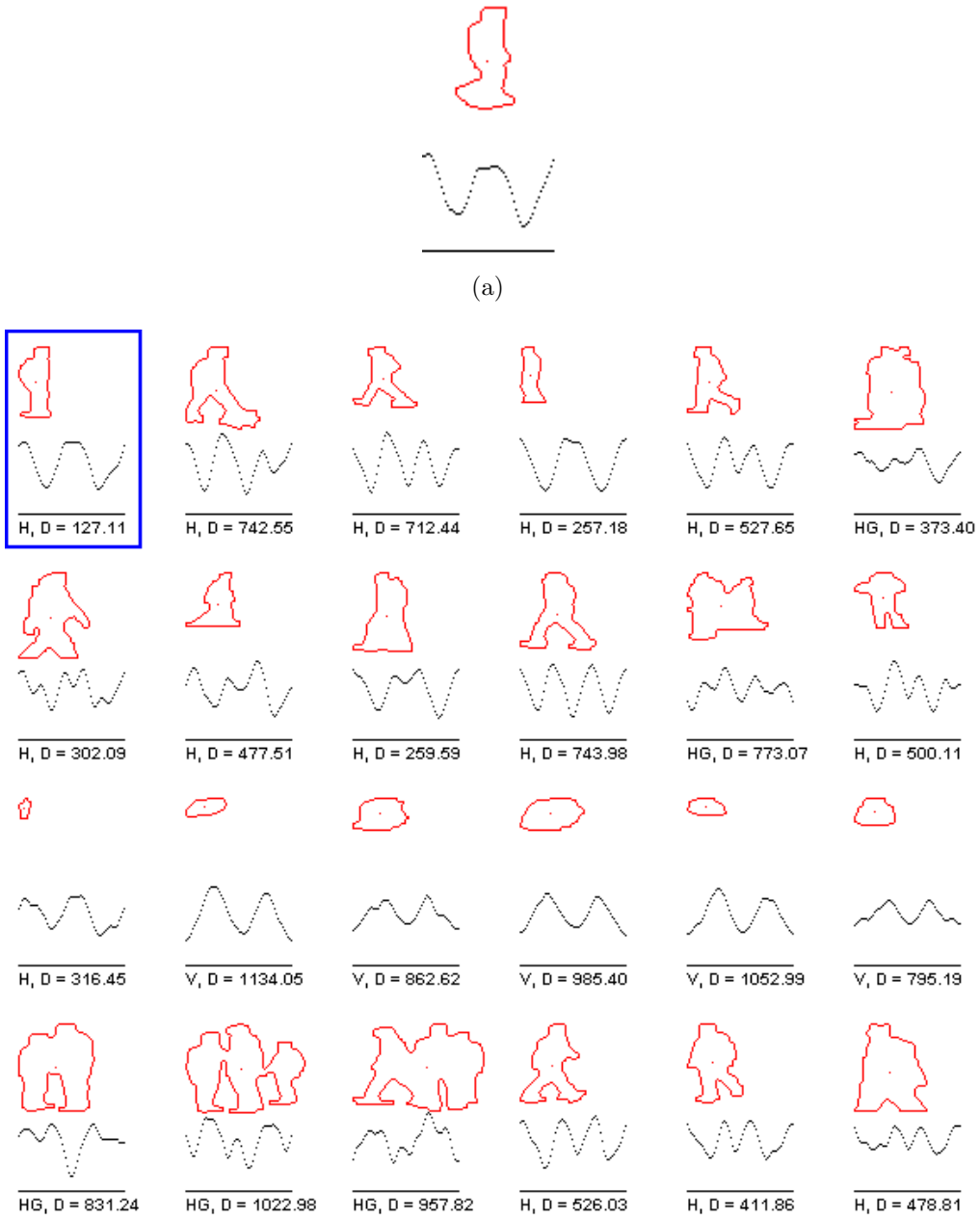


Figure 4.4: Object classification sample. (a) Sample query object (b) Template database objects with distance signals. The type of each object (H: Human, HG: Human Group, V: Vehicle) and the distance (D) between the query object and each database object are shown below the objects.

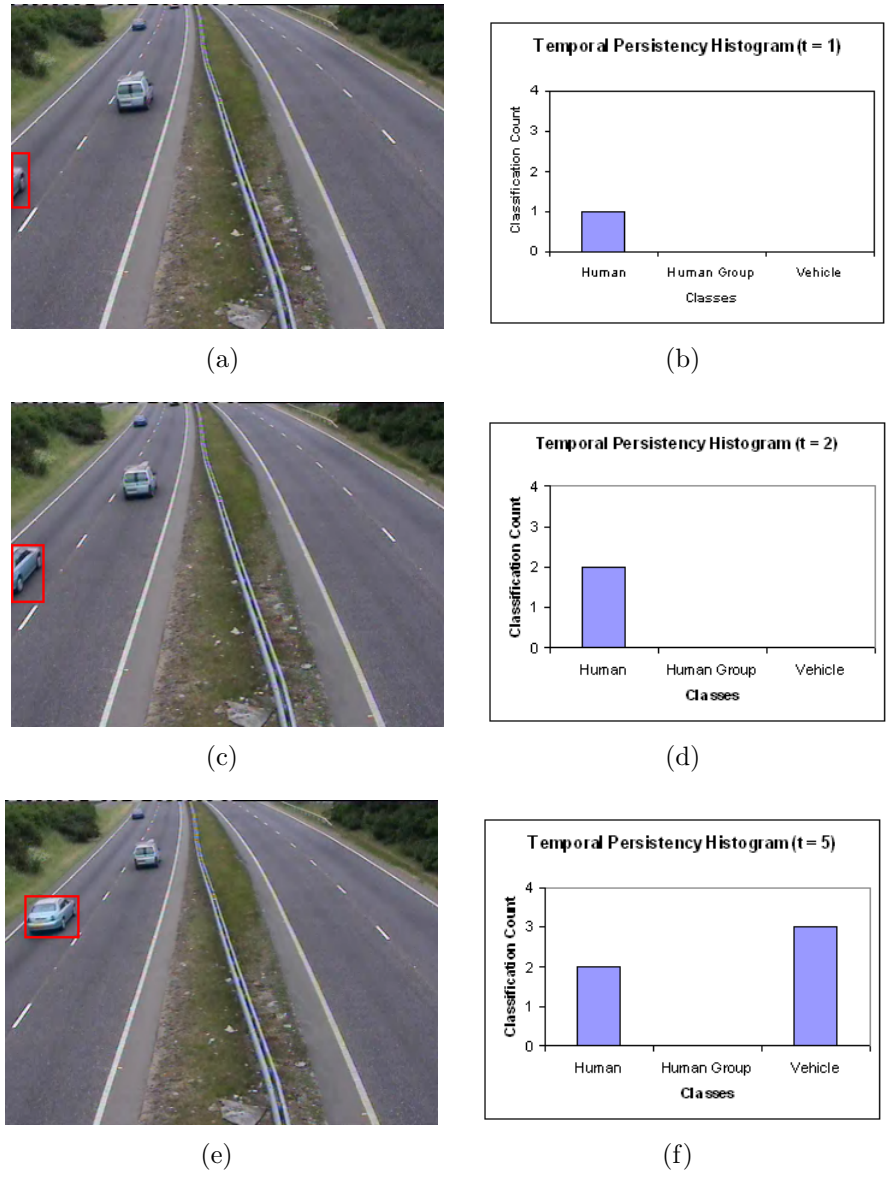


Figure 4.5: Object type histogram for a sample detected object. ((a), (c), (e)) Detected object ((b), (d), (f)) Corresponding object type histogram

Chapter 5

Fire Detection

Surveillance systems are used not only to detect vandal actions performed by humans but also to detect destructive events such as fire to protect security sensitive areas. Traditionally point smoke and fire sensors which sense the presence of certain particles generated by smoke and fire by ionisation or photometry, were used to detect fire. Conventional sensors only aim to sense particles, thus, an important weakness of point detectors is that they are distance limited and fail in open or large spaces.

The strength of using video in fire detection is the ability to serve large and open spaces as well as indoor environments. Current fire and flame detection algorithms are based on the use of color and motion information in video [23, 30]. One weakness encountered in [23] is that fire-like colored moving objects or objects moving in front-of fire-like colored backgrounds are detected as fire regions for short periods of time which lead to false alarms. In our study, which is inspired from the work presented in [23], we not only detect fire and flame colored regions but also analyze the motion in detail to reduce false alarm rates. It is well-known that turbulent flames flicker. Therefore, fire detection scheme can be made more robust by detecting periodic and spatial high-frequency behavior in flame colored pixels compared to existing fire detection systems.

Our fire detection scheme consists of six steps which are depicted in Figure 5.1

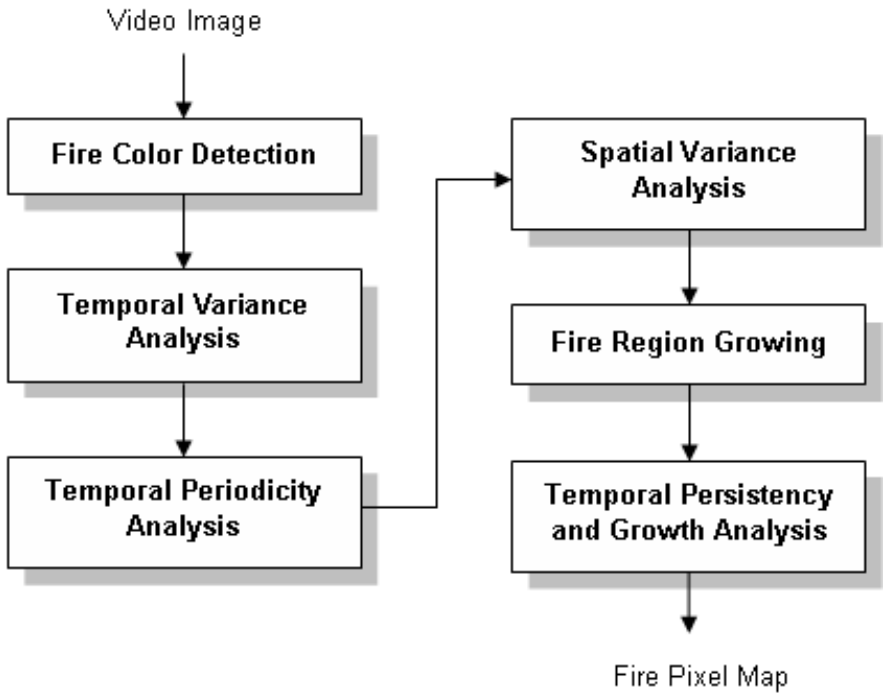


Figure 5.1: The fire detection system diagram.

and briefly listed below:

1. *Detecting fire colored pixels*: Fire colored pixels in an image is detected by using a pre-computed fire color probability distribution.

2. *Temporal variance analysis*: Fire regions exhibit fluctuating intensity changes thus generate high temporal variance. Fire colored rigid objects generally do not cause high temporal variance.

3. *Temporal periodicity analysis*: In some cases, fire colored regions may exhibit high temporal variance. By checking the oscillatory fire color fluctuation, we better distinguish fire regions from ordinary object regions.

4. *Spatial variance analysis*: Fire regions not only generate temporal variance but they also exhibit high spatial variance. In this step spatial variance of possible fire regions are checked to eliminate false alarms.

5. *Fire region growing*: The above checks may filter out true fire pixels. In order to find the exact fire region, we grow the output of the previous steps by using the fire color distribution.

6. *Temporal persistency and growth checks*: Despite all of the checks performed in previous steps, false detection may occur. We eliminate these false alarms by checking the persistency of fire regions and their growth since uncontrolled fire regions grow in time.

The steps *1*, *2* and *5* are similar to the approach presented in [23] where as the steps *3*, *4* and *6* are novel extensions to reduce false alarm rates.

5.1 Color Detection

Generally fire regions in video images have similar colors. This suggests the idea of detecting fire region pixels based on their color values. In order to achieve this, we create a fire color lookup function ($FireColorLookup$) which given an RGB color triple returns whether it is a fire color or not.

The $FireColorLookup$ function uses a fire color predicate formed by several hundreds of fire color values collected from sample images that contain fire regions. These color values form a three dimensional point cloud in RGB color space as shown in Figure 5.2. The problem now reduces to represent this fire color cloud in RGB color space effectively and deciding on the type of a given pixel color by checking whether it is inside this fire color cloud or not.

We decided to represent the fire color cloud by using a mixture of Gaussians in RGB color space. We used the idea presented in [44]. In this approach, the sample set of fire colors $FC = \{c_1, c_2, \ldots, c_n\}$ is considered as a pixel process and a Gaussian mixture model with $N(= 10)$ Gaussian distributions is initialized by using these samples. In other words, we represent the point cloud of fire colored pixels in RGB space by using N spheres whose union almost covers the point cloud. Figure 5.2 shows the sample fire color cloud and the Gaussian distributions

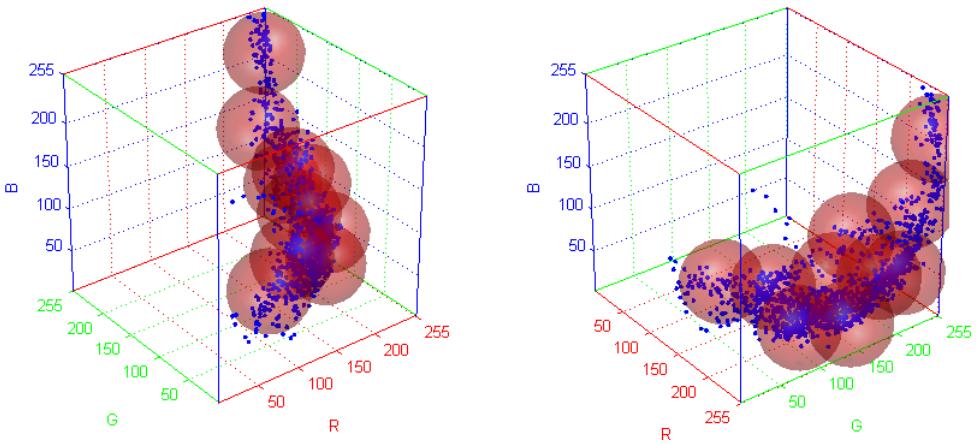


Figure 5.2: Sample fire color cloud in RGB space and Gaussian distribution spheres which cover the cloud shown from two different views.

as spheres which cover the cloud. For a query color c, $FireColorLookup$ function then checks whether any of the Gaussian spheres include the point corresponding to c or not and classifies c as fire or non-fire.

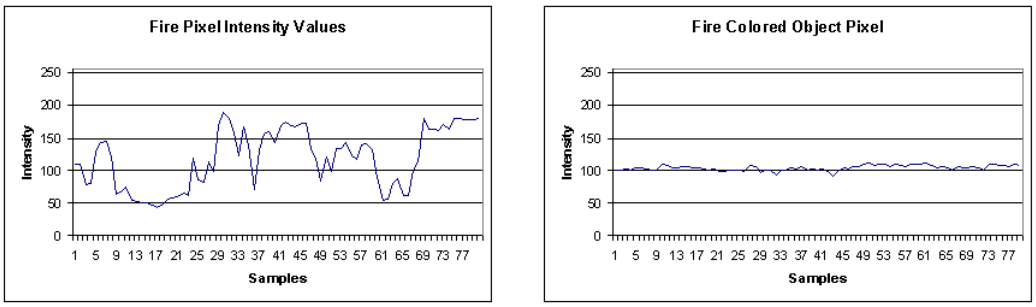
Fire is gaseous and therefore it may become transparent and undetected by our color predicate. Therefore, it is necessary to average the fire color estimate over small windows of time as suggested in [23] as follows:

$$FireColorProb(x) = \frac{\sum_{i=1}^{n} FireColorLookup(I_i(x))}{n} \tag{5.1}$$

$$FireColored(x) = FireColorProb(x) > k_1 \tag{5.2}$$

where n is the total number of images in the subset and I_i is the i^{th} image in the subset, $I_i(x)$ is the RGB color value of the pixel at position x and $k_1(\approx 0.2)$ is an experimentally determined constant. $FireColorProb$ returns a value between zero and one which specifies the probability of pixel at position x being fire. $FireColored$ is a boolean predicate which uses the probability information to mark a pixel as either fire colored or not.

The output of the first step of the algorithm is a binary pixel map $Fire(x)$ that is generated by using $FireColored$ for each pixel position x in the image I.



(a) (b)

Figure 5.3: Temporal variance of the intensity of fire colored pixels. (a) A pixel in true fire region (b) A pixel of a fire colored object

5.2 Temporal Variance Analysis

Color alone is not sufficient to categorize a pixel as part of fire. Ordinary objects, such as a human with fire-colored clothes might be detected as fire if we use color alone. Another distinct property of fire regions is that the flicker of fire causes the pixel intensity values in fire region to fluctuate in time. Figure 5.3 shows the change of intensity values of two different fire colored pixels. The intensity change of true fire colored pixels shows a big variance whereas non-fire objects' pixels show less intensity variation. In order to make use of this feature, we calculate the temporal variance of the intensity values of each fire colored pixel over a small window. The pixels that do not show high frequency behavior are eliminated in this step.

Thinking of global temporal variance in image sequences due to for instance camera noise, we calculate a normalized temporal variance for fire pixels by taking the global non-fire pixels' temporal variance into account as follows [23]:

$$FireDiff\,(x) = Diff\,(x) - AverageNonFireDiff \tag{5.3}$$

where *Diff* and *AverageNonFireDiff* are calculated as follows:

$$Diff\,(x) = \frac{\sum_{i=2}^{n} |G(I_i(x)) - G(I_{i-1}(x))|}{n-1} \tag{5.4}$$

$$AverageNonFireDiff = \frac{\sum_{x,FireColored(x)=0} Diff\,(x)}{\sum_{x,FireColored(x)=0} 1} \tag{5.5}$$

where G is a function given an RGB color returns its gray-level intensity value, *AverageNonFireDiff* is the average variance of the non-fire pixels intensity values, *Diff* is the intensity variance of a pixel, and *FireDiff* is the normalized intensity variance of a fire pixel. Pixels for which $FireDiff\,(x) < k_2 (\approx 15)$ are eliminated from the binary fire pixel map, $Fire(x)$, that is generated by using color predicate in previous step.

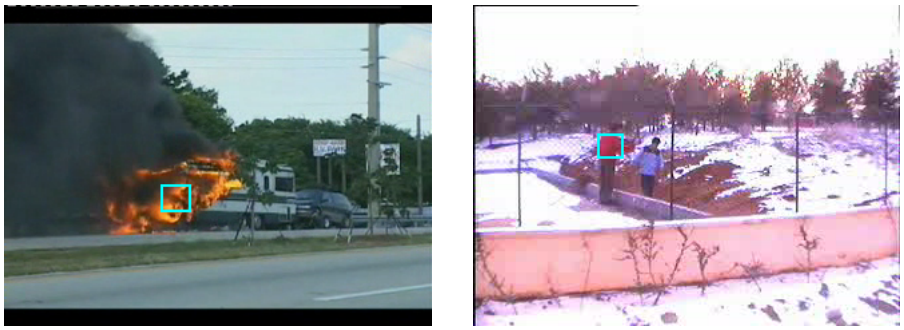
5.3 Temporal Periodicity Analysis

The previous step may fail in some cases, for instance, if the background scene is fire colored, and an object moves in front of it, some of the pixels will be classified as fire for short periods of time since a) the fire color probability would hold b) the object's motion would generate a big temporal intensity variance. In order to eliminate such cases, we look at the oscillatory behavior of the $FireColorLookup$ of a pixel over a small window of time. It is well-known that turbulent flames flicker which significantly increase the frequency content. In other words, a pixel especially at the edge of a flame could appear and disappear several times in one second of a video. The appearance of an object where the $FireColorLookup$ oscillate at a high frequency is a sign of the possible presence of flames.

We calculate the frequency of oscillation of $FireColorLookup$ as follows:

$$TemporalFreq = \frac{\sum_{i=2}^{n} |FireColorLookup_i(x) - FireColorLookup_{i-1}(x)|}{2} \tag{5.6}$$

For true fire pixels $TemporalFreq$ is greater than k_3 Hz, where k_3 is an experimentally determined constant (≈ 3 Hz for a video recorded in 10 Hz). The pixels that have smaller frequency are eliminated from the fire pixel map $Fire(x)$.



(a) (b)

Figure 5.4: Spatial variance of the intensity of pixels. (a) A true fire region $Variance = 1598$ (b) A fire colored object region $Variance = 397$

5.4 Spatial Variance Analysis

Another characteristic of fire regions is that they exhibit larger spatial variance compared to fire colored ordinary objects. Figure 5.4 shows a fire region and a fire colored object and their corresponding spatial color intensity variances.

In order to calculate the spatial variance of fire regions, we first find the isolated fire regions. This is accomplished by applying connected component analysis to the fire pixel map. Let $R = \{p_1, p_2, \ldots, p_n\}$ be a fire region consisting of n pixels. The spatial intensity variance for fire region R is calculated as follows:

$$SpatialMean = \frac{\sum_{i=1}^{n} G(I(p_i))}{n} \tag{5.7}$$

$$SpatialVariance = \frac{\sum_{i=1}^{n} (G(I(p_i)) - SpatialMean)^2}{n} \tag{5.8}$$

For true fire pixels $SpatialVariance$ is greater than k_4, where k_4 is an experimentally determined constant (≈ 500). The pixels belonging to regions that have smaller spatial variance are eliminated from the fire pixel map $Fire(x)$.

5.5 Fire Region Growing

The pixel-level checks applied in previous steps may filter out true fire pixels that do not meet all of the criteria. In order to extract the exact fire region, we grow the output of the previous steps, fire pixel map $Fire(x)$, by using the $FireColorProb$ alone as it is presented in [23]. For a pixel detected as fire, we check its neighboring pixels' $FireColorProb$ values with a smaller threshold and for pixels that pass this check, we set the corresponding entry in the pixel map as fire. The threshold increases as we go far from a fire pixel. The complete fire region growing algorithm is shown in Algorithm 1.

Algorithm 1 Grow fire region

1: $Fire\acute{} \leftarrow Fire$
2: $dist \leftarrow 0$
3: $changed \leftarrow TRUE$
4: **while** ($changed$ = TRUE) **do**
5: $changed \leftarrow FALSE$
6: **for all** pixels $\acute{x}$ that are eight-neighbors of pixels x such that $Fire(x) = 1$ **do**
7: **if** $FireColorProb(\acute{x}) > (k_5 + dist)$ **then**
8: $Fire\acute{}(\acute{x}) \leftarrow 1$
9: $changed \leftarrow TRUE$
10: **end if**
11: **end for**
12: $Fire \leftarrow Fire\acute{}$
13: $dist \leftarrow dist + k_6$
14: **end while**

5.6 Temporal Persistency and Growth Checks

Due to the pixel level checks applied to the image sequence, some false fire regions can be detected for short periods of time. In order to filter out these false alarms, we check the temporal persistency and growth of these fire regions.

In order to check the temporal persistency, we require a fire region to show itself at least in t_1 consecutive frames. We perform this check by using a grid G

of size $(m \times n)$ overlayed on the image. Each cell in this grid has $ImageWidth/m$ width and $ImageHeight/n$ height and has a counter to count number of frames a fire region is seen in this grid cell. For each fire region detected in each frame, we increase the counters of the overlapping cells with 2 and we decrease the counters of non-overlapping cells with 1. In this way, the counters of the cells containing fire will increase rapidly and if the counter exceeds a threshold we flag the first fire possible event, FPE_1.

We also associate the size information of each fire region and check the overlapping areas of a fire region between consecutive frames. If the overlapping area increases we set the second flag to indicate that fire event is possible FPE_2.

If these two conditions (FPE_1 and FPE_2) are met, we are sure that there is fire in the scene and alert the operator.

Chapter 6

Experimental Results

In this chapter we present the test environment and the experimental results of our algorithms.

6.1 Test Application and System

We implemented a video player application (vPlayer) to test our algorithms. The video player can play video clips stored in compressed and uncompressed AVI format. The player application both displays the video data on the screen and at the same time it feeds the image to our video analyzer algorithms such as object tracker or fire detector. The architecture of the player application is made flexible in order to load different types of video clips and use different video analyzers. Thus, we created two APIs: VideoDecompressorAPI to load video images and VideoAnalyzerAPI to analyze video images by using several algorithms. The application is implemented using Microsoft Visual C++ and the decompressors and analyzers are implemented as Dynamic Link Libraries that can be plugged into the player application.

All of the tests in the next sections are performed by using the player application, vPlayer on Microsoft Windows XP Professional operating system on a

Detection Algorithm	Average time to process a frame
Adaptive Background Subtraction	3 msec.
Temporal Differencing	3 msec.
Adaptive Background Mixture Models	15 msec.

Table 6.1: Performance of object detection algorithms

computer with an Intel PIV-2600 MHz CPU and 512 MB of RAM.

6.2 Object Detection and Tracking

We tested the computational performance and detection quality of three different object detection algorithms adaptive background subtraction, temporal differencing and adaptive background mixture models. The time performance analysis, which is the per-frame processing time of these algorithms for an image size of (160×120) is shown in Table 6.1.

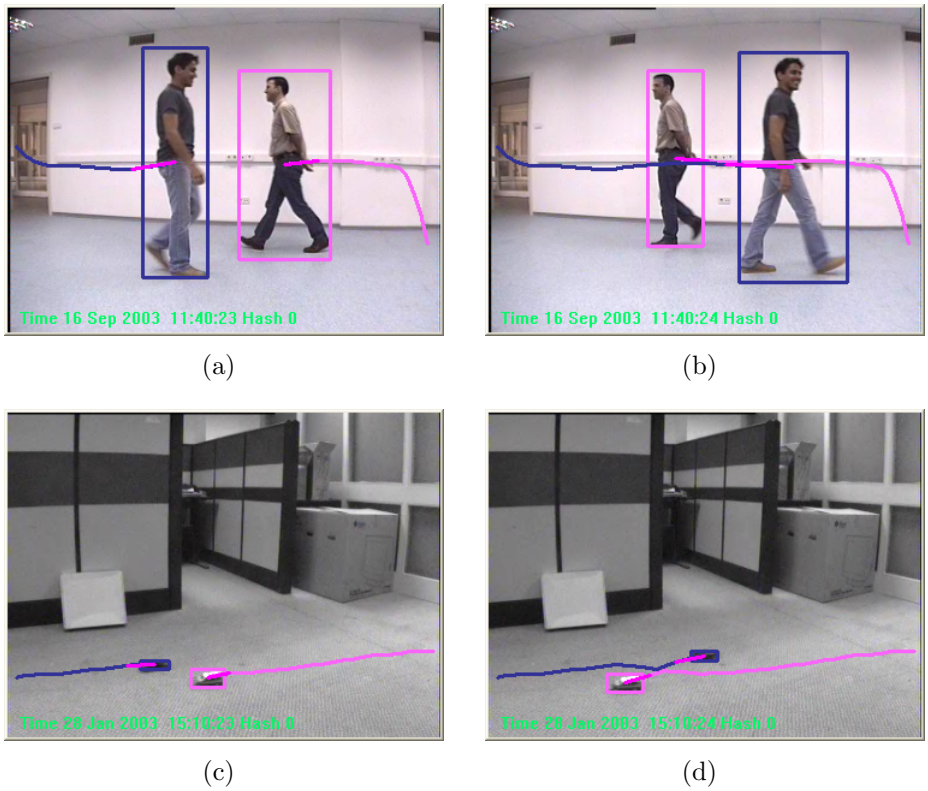
In order to test the performance of the object tracking we used sample indoor and outdoor video clips. We especially tested the performance of the occlusion handling approach presented in Subsection 3.2.2. Table 6.2 shows the number of true occlusions and the number of cases our occlusion handling algorithm identifies the objects correctly after the split and the error rates. Figure 6.1 shows sample tracking scenes where the occlusions are handled successfully.

6.3 Object Classification

In order to test the object classification algorithm we first created a sample object template database by using an application to extract and label object silhouettes. The GUI of this application is shown in Figure 6.2. We used four sample video clips that contain human, human group and vehicle. The number of objects of

Video Clip	# Occlusions	# Successful Occlusion Handling	Success Rate (%)
Movie 1	2	2	100
Movie 2	1	1	100
Movie 3	4	2	50
Movie 4	3	2	67
Movie 5	3	3	100
Movie 6	5	3	60
Total	18	11	75

Table 6.2: Occlusion handling results for sample clips



(a) (b)

(c) (d)

Figure 6.1: Sample video frames before and after occlusions.

Video Clip	# Humans	# Human Groups	# Vehicles
Movie 1	0	0	4
Movie 2	1	1	2
Movie 3	3	2	0
Movie 4	5	3	0
Total	9	7	6

Table 6.3: Number of object types in the sample object template database

	Human	Human Group	Vehicle	Correct (%)
Human	175	13	20	84
Human Group	12	52	14	66
Vehicle	38	22	238	79

Table 6.4: Confusion matrix for object classification

different types extracted from each movie clip is shown in Table 6.3.

We used the sample object database to classify object in several movie clips containing human, human group and vehicle. We prepared a confusion matrix to measure the performance of our object classification algorithm. The confusion matrix is shown in Table 6.4.

6.4 Fire Detection

We compared the performance of our fire detection algorithm (Method1) with the method presented in [23] (Method2). The experimental results show that our algorithm has significantly reduced the false alarms. The results of the comparison are shown in Table 6.5.

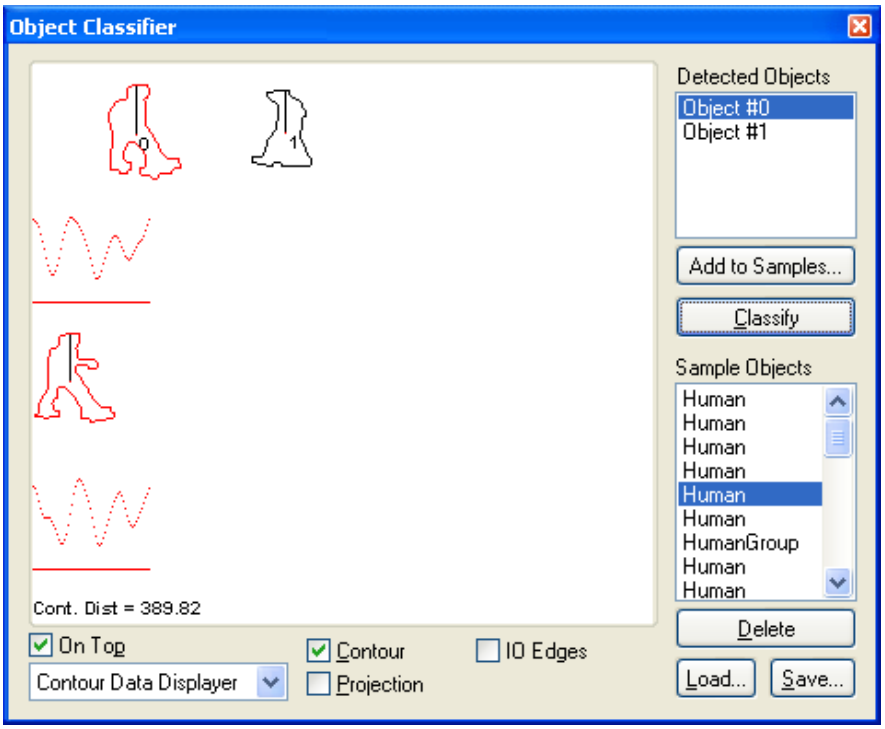


Figure 6.2: The GUI of object template database creation application.

Video Sequences	# Shots with Fire	Method	# Shots detected as Fire	# False+ Frames	Description
Movie 1	0	Method1	0	0	A fire-colored moving truck
		Method2	5	46	
Movie 2	5	Method1	5	0	Fire in a garden
		Method2	5	0	
Movie 3	0	Method1	0	0	A car leaving a fire-colored parking lot
		Method2	1	1	
Movie 4	3	Method1	3	0	A burning box
		Method2	3	7	
Movie 5	8	Method1	8	0	A burning pile of woods
		Method2	10	24	
Movie 6	4	Method1	4	0	Fire behind a man with a fire colored shirt
		Method2	5	15	
Movie 7	0	Method1	0	0	Three men walking in a room
		Method2	2	4	
Movie 8	2	Method1	2	0	Fire in a fireplace
		Method2	2	0	
Movie 9	0	Method1	0	0	A crowded parking lot
		Method2	1	5	
Movie 10	0	Method1	0	0	Traffic on a highway
		Method2	0	0	

Table 6.5: Fire detection performance comparison

Chapter 7

Conclusion and Future Work

In this thesis we presented a set of methods and tools for a "smart" visual surveillance system.

We implemented three different object detection algorithms and compared their detection quality and time-performance. The adaptive background subtraction scheme gives the most promising results in terms of detection quality and computational complexity to be used in a real-time surveillance system with more than a dozen cameras. However, no object detection algorithm is perfect, so is our method since it needs improvements in handling darker shadows, sudden illumination changes and object occlusions. Higher level semantic extraction steps would be used to support object detection step to enhance its results and eliminate inaccurate segmentation.

The proposed whole-body object tracking algorithm successfully tracks objects in consecutive frames. Our tests in sample applications show that using nearest neighbor matching scheme gives promising results and no complicated methods are necessary for whole-body tracking of objects. Also, in handling simple object occlusions, our histogram-based correspondence matching approach recognizes the identities of objects entered into an occlusion successfully after a split. However, due to the nature of the heuristic we use, our occlusion handling algorithm would fail in distinguishing occluding objects if they are of the same

size and color. Also, in crowded scenes handling occlusions becomes infeasible with such an approach, thus a pixel-based method, like optical flow is required to identify object segments accurately.

We proposed a novel object classification algorithm based on the object shape similarity. The method is generic and can be applied to different classification problems as well. Although this algorithm gives promising results in categorizing object types, it has two drawbacks: (a) the method requires effort to create a labeled template object database (b) the method is view dependent. If we could have eliminated (b), the first problem would automatically disappear since one global template database would suffice to classify objects. One way to achieve this may be generating a template database for all possible silhouettes of different classes. This would increase the computational time, but may help to overcome the need for creating a template database for each camera position separately.

The fire detection algorithm we presented is based on an existing method[23] but contains novel extensions which reduces the false alarm rates considerably compared to the method discussed in [23]. Especially, checking the fire colored regions for temporal periodicity and spatial variance and using persistency of fire regions to raise alarms are the novel parts of our method which increases the overall reliability of the fire detection system. The system can be made more robust by incorporating different fire color spectrums and fusion of thermal images.

In short, the methods we presented for "smart" visual surveillance show promising results and can be both used as part of a real-time surveillance system or utilized as a base for more advanced research such as activity analysis in video.

Bibliography

[1] J.K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, March 1999.

[2] A. Amer. Voting-based simultaneous tracking of multiple video objects. In *Proc. SPIE Int. Symposium on Electronic Imaging*, pages 500–511, Santa Clara, USA, January 2003.

[3] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 13:209–216, 1991.

[4] R. Bodor, B. Jackson, and N. Papanikolopoulos. Vision-based human tracking and activity recognition. In *Proc. of the 11th Mediterranean Conf. on Control and Automation*, June 2003.

[5] A. Cavallaro and F. Ziliani. *Image Analysis for Advanced Video Surveillance*, chapter 2.3, pages 57–67. Multimedia Video-Based Surveillance Systems. Kluwer Academic Publishers, Boston, 2000.

[6] H.T. Chen, H.H. Lin, and T.L. Liu. Multi-object tracking using dynamical graph matching. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 210–217, 2001.

[7] R. T. Collins, R. Gross, and J. Shi. Silhouette-based human identification from body shape and gait. In *Proc. of Fifth IEEE Conf. on Automatic Face and Gesture Recognition*, pages 366–371, 2002.

[8] R. Cutler and L.S. Davis. Robust real-time periodic motion detection, analysis and applications. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 781–796, 2000.

[9] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using non-parametric kernel density estimation for visual surveillance. In *Proc. of the IEEE*, volume 90, July 2002.

[10] R. T. Collins et al. A system for video surveillance and monitoring: VSAM final report. Technical report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, May 2000.

[11] T. Brodsky et al. *Visual Surveillance in Retail Stores and in the Home*, chapter 4, pages 51–61. Video-Based Surveillance Systems. Kluwer Academic Publishers, Boston, 2002.

[12] H. Fujiyoshi and A.J. Lipton. Real time human motion analysis by image skeletonization. In *Proc. of Workshop Applications of Computer Vision*, pages 15–21, 1998.

[13] D. M. Gavrila. The analysis of human motion and its application for visual surveillance. In *Proc. of the 2nd IEEE International Workshop on Visual Surveillance*, pages 3–5, Fort Collins, U.S.A., 1999.

[14] D. Greenhill, P. Remagnino, and G. A. Jones. *VIGILANT*, chapter 16, pages 193–204. Video-Based Surveillance Systems. Kluwer Academic Publishers, Boston, 2002.

[15] I. Haritaoglu. *A Real Time System for Detection and Tracking of People and Recognizing Their Activities*. PhD thesis, University of Maryland at College Park, 1998.

[16] I. Haritaoglu, R. Cutler, D. Harwood, and L. S. Davis. Backpack: Detection of people carrying objects using silhouettes. *Computer Vision and Image Understanding*, 81(3):385–397, 2001.

[17] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: A real time system for detecting and tracking people. In *Computer Vision and Pattern Recognition*, pages 962–967, 1998.

[18] G. Healey, D. Slater, T. Lin, B. Drda, and D. Goedeke. A system for real-time fire detection. *Computer Vision and Pattern Recognition*, pages 605–606, 1993.

[19] F. Heijden. *Image Based Measurement Systems: Object Recognition and Parameter Estimation*. Wiley, January 1996.

[20] J. Heikkila and O. Silven. A real-time system for monitoring of cyclists and pedestrians. In *Proc. of Second IEEE Workshop on Visual Surveillance*, pages 74–81, Fort Collins, Colorado, June 1999.

[21] T. Horprasert, D. Harwood, and L.S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. of IEEE Frame Rate Workshop*, pages 1–19, Kerkyra, Greece, 1999.

[22] H.Ramoser, T.Schlgl, M.Winter, and H.Bischof. Shape-based detection of humans for video surveillance. In *Proc. of IEEE Int. Conf. on Image Processing*, Barcelona, Spain, 2003.

[23] W. Phillips III, M. Shah, and N. Da Vitoria Lobo. Flame recognition in video. In *Fifth IEEE Workshop on Applications of Computer Vision*, pages 224–229, December 2000.

[24] Y. Ivanov, C. Stauffer, A. Bobick, and W.E.L. Grimson. Video surveillance of interactions. In *International Workshop on Visual Surveillance*, pages 82–89, Fort Collins, Colorado, June 1999.

[25] S. Ju, M. Black, and Y. Yaccob. Cardboard people: a parameterized model of articulated image motion. In *Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

[26] P. KaewTraKulPong and R. Bowden. *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection*, chapter 11,

pages 135–144. Video-Based Surveillance Systems. Kluwer Academic Publishers, Boston, 2002.

[27] S. Khan and M. Shah. Tracking people in presence of occlusion. In *Proc. of Asian Conference on Computer Vision*, pages 1132–1137, Taipei, Taiwan, January 2000.

[28] A. J. Lipton. Local application of optic flow to analyse rigid versus non-rigid motion. Technical Report CMU-RI-TR-99-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1999.

[29] A. J. Lipton, H. Fujiyoshi, and R.S. Patil. Moving target classification and tracking from real-time video. In *Proc. of Workshop Applications of Computer Vision*, pages 129–136, 1998.

[30] C. B. Liu and N. Ahuja. Vision based fire detection. In *IEEE International Conference on Pattern Recognition*, Cambridge, UK, August 2004. to appear.

[31] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, August 1998.

[32] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. In *Proc. of Statistical Methods in Video Processing Workshop*, pages 7–12, June 2002.

[33] J. S. Marques, P. M. Jorge, A. J. Abrantes, and J. M. Lemos. Tracking groups of pedestrians in video sequences. In *Proc. of IEEE Workshop on Multi-Object Tracking*, page 101, Madison, June 2003.

[34] A. M. McIvor. Background subtraction techniques. In *Proc. of Image and Vision Computing*, Auckland, New Zealand, 2000.

[35] S.J. McKenna, S. Jabri, Z. Duric, and H. Wechsler. Tracking interacting people. In *Proc. of International Conference on Automatic Face and Gesture Recognition*, pages 348–353, 2000.

[36] F. Oberti, G. Ferrari, and C. S. Regazzoni. *A Comparison between Continuous and Burst, Recognition Driven Transmission Policies in Distributed*

3GSS, chapter 22, pages 267–278. Video-Based Surveillance Systems. Kluwer Academic Publishers, Boston, 2002.

[37] J. Owens and A. Hunter. A fast model-free morphology-based object tracking algorithm. In *Proc. of British Machine Vision Conference*, pages 767–776, Cardiff, UK, September 2002.

[38] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proc. of IEEE Int. Conf. on Intelligent Vehicles*, pages 241–246, Germany, October 1998.

[39] C. Regazzoni and P. Varshney. Multi-sensor surveillance. In *IEEE Int. Conf. Image Processing*, pages 497–500, 2002.

[40] R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *Proc. of IEEE CVPR Workshop on the Interpretation of Visual Motion*, Santa Barbara, CA, 1998.

[41] M. Saptharishi, J.B. Hampshire II, and P. Khosla. Agent-based moving object correspondence using differential discriminative diagnosis. In *Proc. of Computer Vision and Pattern Recognition*, pages 652–658, 2000.

[42] E. Saykol, U. Gudukbay, and O. Ulusoy. A histogram-based approach for object-based query-by-shape-and-color in multimedia databases. Technical Report BUCE-0201, Bilkent University, 2002.

[43] E. Saykol, G. Gulesir, U. Gudukbay, and O. Ulusoy. KiMPA: A kinematics-based method for polygon approximation. In *International Conference on Advances in Information Systems (ADVIS'02)*, pages 186–194, Izmir, Turkey, 2002.

[44] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page 246252, 1999.

[45] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Pattern Recognition and Machine Intelligence*, 22(8):747–757, August 2000.

[46] B. U. Toreyin. Moving object detection and tracking in wavelet compressed video. Master's thesis, Bilkent University, Department of Electrical and Electronics Engineering, 2003.

[47] R.C. Veltkamp and M. Hagedoorn. *State-of-the-art in shape matching*, pages 87–119. Principles of Visual Information Retrieval. Springer, 2001.

[48] H. Wang and S.F. Chang. Automatic face region detection in mpeg video sequences. In *Electronic Imaging and Multimedia Systems*, pages 160–168, SPIE Photonics China, November 1996.

[49] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, March 2003.

[50] L. Wang, H. Ning, T. Tan, and W. Hu. Fusion of static and dynamic features of body biometrics for gait recognition. In *Proc. of International Conference on Computer Vision*, pages 1449–1454, Nice, France, 2003.

[51] L. Wixson and A. Selinger. Classifying moving objects as rigid or non-rigid. In *Proc. of DARPA Image Understanding Workshop*, pages 341–358, 1998.

[52] C. R. Wren, A. Azarbayejani, T. J. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Pattern Recognition and Machine Intelligence*, 19(7):780–785, July 1997.

[53] M. Xu and T. Ellis. *Colour-Invariant Motion Detection under Fast Illumination Changes*, chapter 8, pages 101–111. Video-Based Surveillance Systems. Kluwer Academic Publishers, Boston, 2002.

[54] T. Zhao, R. Nevatia, and F. Lv. Segmentation and tracking of multiple humans in complex situations. In *Proc. of USC Computer Vision*, pages 194–201, 2001.

[55] X. Zhou, R. T. Collins, T. Kanade, and P. Metes. A master-slave system to acquire biometric imagery of humans at distance. In *First ACM SIGMM International Workshop on Video Surveillance*, pages 113–120. ACM Press, 2003.