

# **EVENT DETECTION IN SURVEILLANCE VIDEO**

by

Ricardo Augusto Castellanos Jimenez

A Thesis Submitted to the Faculty of

The College of Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, Florida

May 2010


# EVENT DETECTION IN SURVEILLANCE VIDEO

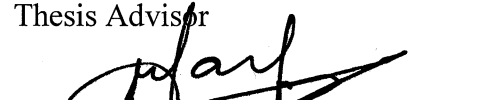
by

Ricardo Augusto Castellanos Jimenez

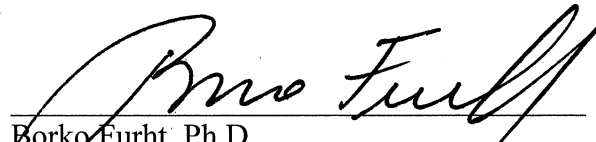
This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Hari Kalva, Department of Computer and Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

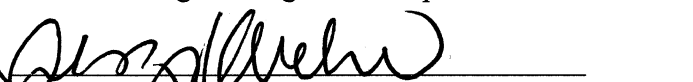
## SUPERVISORY COMMITTEE:

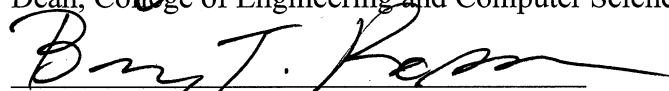
  
Hari Kalva, Ph.D.  
Thesis Advisor

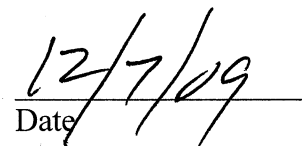
  
Oge Marques, Ph.D.

  
Bassem Alhalabi, Ph.D.

  
Borko Furht, Ph.D.  
Chair, Department of Computer and  
Electrical Engineering and Computer Science

  
Karl K. Stevens, Ph.D., P.E.  
Dean, College of Engineering and Computer Science

  
Barry T. Rosson, Ph.D.  
Dean, Graduate College

  
Date 12/7/09

## **ACKNOWLEDGMENTS**

I would like to express my sincere appreciation to my thesis advisor, Dr. Hari Kalva. His guidance and encouragement made this work possible. I also want to thank to the members of the Thesis committee Dr. Oge Marques and Dr. Bassem Alhalabi. Additionally, I want to thank Dr Maria Petrie for her constant advice and support and my friends and colleagues for all the support and patience during this academic process.

Finally, I want to thank my parents and family. They were the biggest motivation to reach this goal. Their words and advice gave me the strength and guidance in this journey. This accomplishment is not only mine, it belongs to them too.

## **ABSTRACT**

Author: Ricardo Augusto Castellanos Jimenez  
Title: Event Detection in Surveillance Video  
Institution: Florida Atlantic University  
Thesis Advisor: Dr. Hari Kalva  
Degree: Master of Science  
Year: 2010

Digital video is being used widely in a variety of applications such as entertainment, surveillance and security. Large amount of video in surveillance and security requires systems capable to processing video to automatically detect and recognize events to alleviate the load on humans and enable preventive actions when events are detected. The main objective of this work is the analysis of computer vision techniques and algorithms used to perform automatic detection of events in video sequences. This thesis presents a surveillance system based on optical flow and background subtraction concepts to detect events based on a motion analysis, using an event probability zone definition. Advantages, limitations, capabilities and possible solution alternatives are also discussed. The result is a system capable of detecting events of objects moving in opposing direction to a predefined condition or running in the scene, with precision greater than 50% and recall greater than 80%.

# EVENT DETECTION IN SURVEILLANCE VIDEO

<b>TABLES.....</b>	<b>ix</b>
<b>FIGURES.....</b>	<b>x</b>
<b>Chapter 1 INTRODUCTION.....</b>	<b>1</b>
<i>1.1 Overview and Motivation.....</i>	<i>1</i>
<i>1.2 Problem Statement and Objective.....</i>	<i>3</i>
<i>1.3 Context and Scope.....</i>	<i>4</i>
<i>1.4 Main Contributions.....</i>	<i>5</i>
<i>1.5 Overview of the Thesis.....</i>	<i>6</i>
<b>Chapter 2 BACKGROUND AND RELATED WORK.....</b>	<b>7</b>
<i>2.1 Introduction.....</i>	<i>7</i>
<i>2.2 General Framework.....</i>	<i>7</i>
2.2.1 Environment Modeling.....	8
2.2.2 Motion Segmentation.....	9
2.2.3 Object Classification.....	10
2.2.4 Tracking.....	11
2.2.5 Behavior Understanding and Description.....	12
2.2.6 Personal Identification.....	13
2.2.7 Fusion of Information from Different Cameras.....	13
<i>2.3 Theoretical Background.....</i>	<i>14</i>
2.3.1 Foreground and Background Estimation.....	14

2.3.2	Mathematical Morphology.....	18
2.3.3	Optical Flow.....	23
2.3.4	Tracking .....	28
<b>Chapter 3</b>	<b>PROPOSED SOLUTION.....</b>	<b>31</b>
3.1	<i>Introduction.....</i>	<i>31</i>
3.2	<i>General Description.....</i>	<i>31</i>
3.2.1	Event Probability Zone Definition.....	32
3.2.2	Block Diagram .....	33
3.3	<i>Detailed Description.....</i>	<i>34</i>
3.3.1	Pre-processing.....	34
3.3.2	Foreground/Background Estimation.....	37
3.3.3	Optical Flow.....	38
3.3.4	Segmentation and Classification.....	40
3.3.5	Post-processing .....	43
<b>Chapter 4</b>	<b>IMPLEMENTATION .....</b>	<b>46</b>
4.1	<i>Introduction.....</i>	<i>46</i>
4.2	<i>Pre-Processing.....</i>	<i>47</i>
4.2.1	Video Acquisition .....	47
4.2.2	Region of Interest (ROI) Extraction .....	48
4.2.3	Conversion to Grayscale .....	48
4.3	<i>Foreground/Background Estimation .....</i>	<i>49</i>
4.3.1	Frame Difference .....	49
4.3.2	Approximate Median .....	50
4.3.3	Mixture of Gaussians .....	50
4.4	<i>Optical Flow .....</i>	<i>51</i>
4.4.1	Horn-Schunck method .....	51
4.4.2	Lucas-Kanade method .....	52

4.5	<i>Segmentation and Classification</i> .....	52
4.5.1	Segmentation.....	52
4.5.2	Classification and Clustering .....	53
4.6	<i>Post-Processing</i> .....	54
4.6.1	Histogram.....	54
4.6.2	Box Filter .....	54
4.6.3	Thresholding .....	55
4.6.4	Differentiation.....	55
4.6.5	Distance Filter .....	56
<b>Chapter 5</b>	<b>EXPERIMENTS AND RESULTS .....</b>	<b>57</b>
5.1	<i>Introduction</i> .....	57
5.2	<i>Pre-Processing</i> .....	58
5.2.1	Video Acquisition .....	58
5.2.2	Region of Interest (ROI) Extraction .....	60
5.2.3	Conversion to Grayscale .....	61
5.3	<i>Foreground/Background Estimation</i> .....	62
5.3.1	Frame Difference .....	62
5.3.2	Approximate Median .....	62
5.3.3	Mixture of Gaussians .....	63
5.4	<i>Optical Flow</i> .....	64
5.5	<i>Segmentation and Classification</i> .....	66
5.5.1	Segmentation.....	66
5.5.2	Classification and Clustering .....	67
5.6	<i>Post-Processing</i> .....	68
5.7	<i>Statistics</i> .....	71
5.7.1	Opposing Flow event .....	72
5.7.2	Person Runs event.....	76

5.8 <i>Summary</i> .....	80
<b>Chapter 6 CONCLUSIONS AND FUTURE WORK</b> .....	<b>82</b>
6.1 <i>Conclusions</i> .....	82
6.2 <i>Future Work</i> .....	85
<b>BIBLIOGRAPHY</b> .....	<b>87</b>



## TABLES

Table 1: Event descriptions.....	32
Table 2: Set of videos and length for OpposingFlow event detection.....	59
Table 3: Set of videos and length for PersonRuns event detection.....	60
Table 4: Time analysis for OpposingFlow events .....	72
Table 5: Time analysis to process a frame in OpposingFlow events.....	73
Table 6: Detections for OpposingFlow events.....	74
Table 7: Time analysis for PersonRuns events .....	76
Table 8: Time analysis to process a frame in PersonRuns events .....	77
Table 9: Detections for PersonRuns events .....	78
Table 10: Byte overhead for MPEG-7 Descriptors.....	86

## FIGURES

Figure 1. General framework of a visual surveillance system.....	8
Figure 2. Background subtraction process.....	15
Figure 3: Structuring element models.....	19
Figure 4: Erosion example.....	20
Figure 5: Dilation example .....	21
Figure 6: Opening example.....	22
Figure 7: Closing example.....	23
Figure 8: Optical flow and motion field of a barber's pole .....	24
Figure 9: Taxonomy of tracking parameters.....	30
Figure 10: Event probability model .....	33
Figure 11: General Block Diagram.....	34
Figure 12: Pre-processing block diagram .....	35
Figure 13: FG/BG Estimation block diagram.....	37
Figure 14: Optical Flow block diagram.....	39
Figure 15: Segmentation and classification block diagram .....	40
Figure 16: Post-Processing block diagram .....	43
Figure 17: Groups of Videos.....	57
Figure 18: ROI selection and extraction .....	61

Figure 19: Threshold comparison for Frame Difference method .....	62
Figure 20: Threshold comparison for Approximate Median method .....	63
Figure 21: Parameter test for Mixture of Gaussians method. ....	63
Figure 22: Horn-Schunk optical flow vectors.....	64
Figure 23: Lucas-Kanade optical flow vectors .....	65
Figure 24: Segmentation process .....	66
Figure 25: Optical flow vectors after segmentation.....	67
Figure 26: Optical flow vectors plot .....	68
Figure 27: Threshold values to optimize the detections percentage .....	69
Figure 28: Post-processing stage .....	70
Figure 29: Detection statistics based on ground truth annotations .....	71
Figure 30: OpposingFlow Recall .....	75
Figure 31: OpposingFlow Precision .....	75
Figure 32: Threshold impact for OpposingFlow .....	76
Figure 33: PersonRuns Recall.....	78
Figure 34: PersonRuns Precision .....	79
Figure 35: Threshold impact for PersonRuns .....	80

## **Chapter 1 INTRODUCTION**

### **1.1 Overview and Motivation**

The huge accumulation of digital data in this new century has become an interesting circumstance where storage and processing of such quantities of information are the key factors to satisfy user requirements and expectations. Multimedia data such as video sequences in visual surveillance systems is a very important topic and probably one of the most illustrative examples of this circumstance because the large demand for analysis and synthesis that is needed to understand the contents to determine specific actions based on registered events. Events are phenomena or circumstances that happen at a given place and time which can be identified without ambiguities, for example, a person entering in a forbidden place, a suspicious object abandoned in a public place or a car parking in a garage.

Digital video recording devices are now ubiquitous and pervasive in our daily lives. They are mounted indoors and outdoors everywhere: offices, rooms, halls, banks, hotels, hospitals, casinos, airports, parking lots, buildings, military sites, streets and intersections; some vehicles even have cameras recording passengers and the surroundings of the car. The wide range of potential applications includes[2]:

- Access control in special areas.
- Person-specific identification.
- Crowd flux statistics and congestion analysis.
- Anomaly detection and alarming.
- Interactive surveillance using multiple cameras.

Computer Vision (CV) technologies are intended to perform intelligent tasks with these “digital eyes”, attaching “brains” to the imaging devices and thus, creating a very useful tool used for video surveillance, entertainment/augmented reality applications, autonomous vehicles and driver assistance systems, robotics and smart health care.

Visual Surveillance systems, which is the main scope for this work, address real-time observation of objects in some environment leading to a description about the activities or interaction of the objects within the environment or among the objects. However, a human operator has either to watch a massive amount of video data in real-time with full attention to detect any anomalies or events, or the video data can only be used as evidence after the abnormal event has occurred, due to the lack of real-time automatic tracking and analysis.

An Automatic Video Surveillance system comprises different functional blocks such as foreground segmentation, object detection/tracking, human or object analysis and finally, activity or behavioral analysis. These blocks are implemented using Computer Vision

techniques and algorithms alleviating the load on humans and enabling preventative acts or alarms when a specific event is detected.

## **1.2 Problem Statement and Objective**

Traditional Video Surveillance systems based on human interaction are ineffective as the number of cameras exceeds the capability of human operators to monitor them. The task is time expensive for the human operator demanding full attention in real time to detect events and leading to low accuracy in detections due to fatigue, lack of observation or lapse in concentration.

Another limitation entails the case when there are not human operators monitoring cameras in real time and the video is recorded to be analyzed afterward if any abnormality is reported (i.e. theft, murder, etc) to be used as an evidence.

This work investigates systems capable to detect events automatically in video surveillance applications reducing or suppressing human interaction with the system and reporting alerts based on the events detected.

### 1.3 Context and Scope

This work has been motivated by the Event Detection in Airport Surveillance task organized by the Text REtrieval Conference (TRECVID). The TRECVID evaluation meetings are on-going series of workshops focusing on a list of different information retrieval research areas in content based retrieval of video. TRECVID is co-sponsored by the National Institute of Standards and Technology (NIST) and the Intelligence Advanced Projects Activity (IARPA) of the United States Office of the Director of National Intelligence (ODNI).

The goal of the TRECVID evaluation is to build and evaluate systems that can detect instances of a variety of observable events in the airport surveillance domain based on video surveillance data collected by the UK Home Office at the London Gatwick International Airport.

This thesis focuses on the combination of functional blocks using computer vision techniques to detect and identify events based on motion analysis. “OpposingFlow” and “PersonRuns” events described in the TRECVID Event Annotation guidelines[25] were selected, because they have common characteristics to be used with the proposed solution such as prolonged motion and object’s displacement. Moreover, these two events were common to the Camera1 video subset, sharing the same environmental conditions.

## 1.4 Main Contributions

The following are the main contributions of this work:

- Implementation of an event detection module combining different image processing and computer vision techniques to detect “OpposingFlow” and “PersonRuns” events.
- Implementation and comparison of different background subtraction techniques with the goal of extracting relevant/moving objects from video frames.
- Implementation and comparison of optical flow techniques to analyze the motion estimation of the objects moving in video sequences and determine the predominant direction of movement.
- Establishment of a framework for students and researchers who may continue working on related topics of activity identification and behavior analysis.
- Implementation of a modular video surveillance system to evaluate and combine diverse algorithms for event detection.



## **1.5 Overview of the Thesis**

The remaining chapters of this thesis are structured as follows: Chapter 2 provides background information on visual surveillance systems and algorithms, Chapter 3 describes in detail the proposed solution where the optical flow algorithm is the main core of the system to determine the direction of movement, Chapter 4 presents the implementation of the system and the description of the algorithms used for this approach, Chapter 5 contains the experiments and results based on the implementation, and Chapter 6 presents conclusions and possibilities for future work.

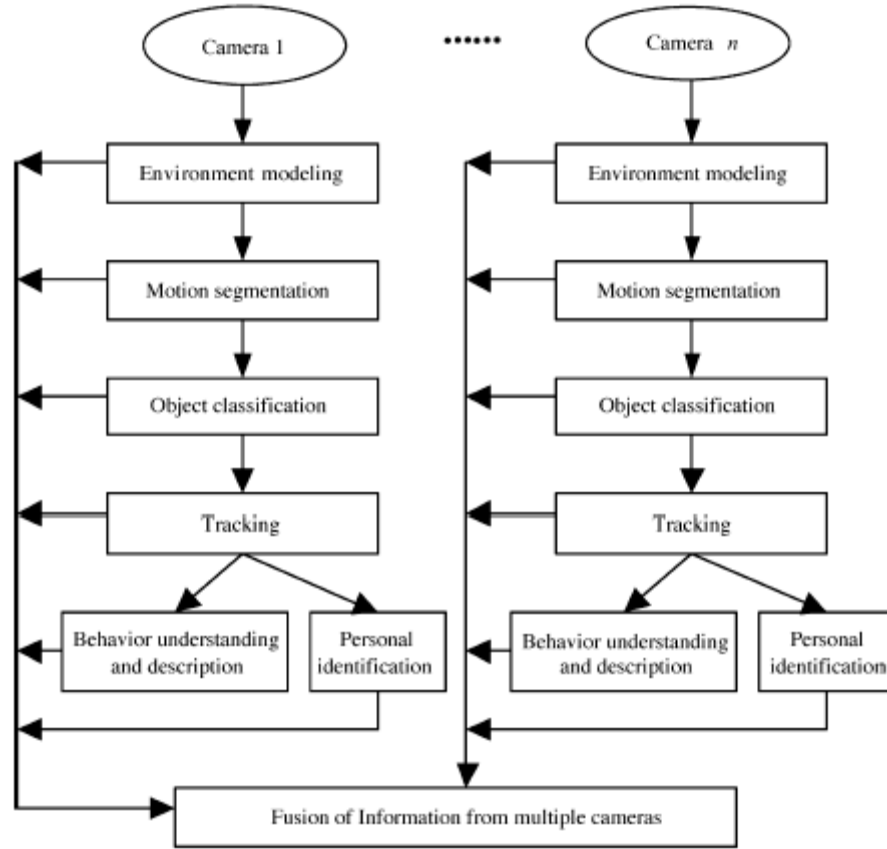
## **Chapter 2 BACKGROUND AND RELATED WORK**

### **2.1 Introduction**

In this chapter we review the principal components of an automatic video surveillances system as an introduction to the general concepts and algorithms associated with this thesis.

### **2.2 General Framework**

The input to a video surveillance system is video streams coming from a single or multiple cameras. The system analyzes the video content going through each single block separating the foreground from the background, detecting and tracking the objects, and performing a high-level analysis. The high-level analysis provides results such as a scenario being *normal* or *abnormal* and based on this result, the system can report the state of the process to facilitate a human operator to focus on the abnormal scenarios without having to stare at the video trying to find any anomaly.



**Figure 1. General framework of a visual surveillance system [2].**

Figure 1 depicts the general framework of a visual surveillance system which is composed of different blocks with specific purposes to accomplish the required system tasks. A brief explanation for each functional block is detailed below[2]:

### 2.2.1 Environment Modeling

A sequence is a set of consecutive frames recorded at the same location. Therefore, group of common elements are shared within this set, and this is what is referred to as *background*. The active construction and updating of the background model is

indispensable to visual surveillance so the next blocks in the pipeline depend on the accuracy of this model.

The background modeling may have different challenges depending on the position of the camera:

- For fixed cameras, the main challenge is to automatically recover and update background images from a dynamic sequence. Different factors such as variations in luminance and shadows may present many difficulties in the acquirement and updating of background images. Some algorithms designed to solve this problems include temporal average of an image sequence[2], adaptive Gaussian estimation[2,4] and parameter estimation based on pixel processes[5].
- For mobile cameras, motion compensation is needed to construct temporary background images[2].

### 2.2.2 Motion Segmentation

The objective is to separate foreground from background in the video sequence. Foreground detection is generally easier in the indoor environment because the outdoor environment is more complex, as wavering tree branches, flickering water surfaces, periodic opening and closing of doors are occurring.

One of the most generalized methods is the background subtraction for motion segmentation used when the environment modeling described before has a relatively static background[3,4]. Moving regions in an image are detected by taking the difference between the current image and the reference background image in a pixel by pixel approach. Other methods, such as temporal differencing, make use of the pixel-wise differences between two or three consecutive frames in an image sequence to extract the moving regions[3], this method is very adaptive to dynamic environments but generally does a poor job of extracting all the relevant pixels. Finally, the optical flow based motion segmentation method uses characteristics of flow vectors of moving objects over time to detect moving regions in an image sequence and can be used to detect independently moving objects even in the presence of camera motion[5,9]. However, this method is computationally complex and has constraints to be applied in real time without specialized hardware[8,9].

### 2.2.3 Object Classification

Once the segmentation process has been completed, it is necessary to perform an object classification in order to identify the different moving regions for further analysis in the system. This task will be very helpful to define the moving regions as moving objects with a higher level of knowledge for tracking purposes or behavior observation and analysis. There are two main categories for classifying moving objects: Shape-based and Motion-based classification.

### 2.2.3.1 Shape-based classification

A wide variety of descriptions of shape information of motion regions such as points, boxes, silhouettes and blobs are available to classify the moving objects. Features such as blob dispersedness, blob area, aspect ratio of the bounding box among others, are of special interest to classify the blobs in different classes. In this way, blobs can be classified as rigid or non-rigid objects, humans, vehicles and so on.

### 2.2.3.2 Motion-based classification

This approach is useful to classify moving objects exploiting the analysis of the periodic motion. Time-frequency analysis is applied to detect and characterize the periodic motion so classification of moving objects is implemented using periodicity. A particular application of this method is used to distinguish human motion from other objects such as vehicles.

### 2.2.4 Tracking

Using the features extracted for the classified objects and their defined characteristics, it is possible to localize its position along the different sequence of frames[7,8,9]. Tracking objects over time typically involves matching them in consecutive images using features

such as points, lines or blobs. Once the object is tracked, very useful information such as position, velocity, centroid and periodicity becomes available and can be used for further processing and analysis but this is only possible when the object has been tracked for a given period of time[5,7].

### 2.2.5 Behavior Understanding and Description

To detect the anomaly of a scene, it is necessary to model the behavior of the objects in the frames. This task can be performed using the information gathered in the previous blocks where the object is recognized and classified using specific features like position, blob area, contour, displacement, direction of movement, magnitude of movement, color, etc. The analysis of these features during time can help us to describe the behavior of the objects as well as the interaction with other objects, based on the changes they are experimenting. Once the behavior is identified, it is translated into high level human expressions to be matched with previous defined patterns.

The use of histograms to model the object's behavior is very helpful to perform this task but other approaches such as finite-state machines[2], neural networks[2,9] and Hidden Markov Models[2,7] are also used.

### 2.2.6 Personal Identification

This block does not apply to all the video surveillance systems, but was included in the general video surveillance framework because its faculty to be used for face recognition purposes. This way, human face and gait is analyzed in order to identify subjects in a non intrusive fashion.

With this approach, the video surveillance system can give answers to the questions involving either what? (related to the object detection) or who? (related to the object identification).

### 2.2.7 Fusion of Information from Different Cameras

Depending on the video surveillance infrastructure, it would be desirable to gather information from different cameras and sensors to expand the surveillance area. Multiple view information can overcome problems such as occlusion and will be very useful for tracking purposes when the object moves among the visual field of different cameras, so the tracking process is not limited to only one camera and can be extended to cover a much bigger area.



The camera fusion involves more complexity in the process and the infrastructure has to deal not only with the camera installation[1,3] but also the camera calibration and synchronization[2,3].

## **2.3 Theoretical Background**

In this section, a theoretical description of algorithms relevant to this work is provided.

### **2.3.1 Foreground and Background Estimation**

This process uses the concepts for Environmental Modeling (Section 2.2.1) and Motion Segmentation (Section 2.2.2) combined into a single block which is usually known as Background Subtraction. Extracting the foreground from the background is an important step in the video surveillance pipeline and represents one of the most common tasks when trying to detect moving objects in video sequences.

The background subtraction scheme works as follows: Each frame is individually compared to a reference background model performing a pixel by pixel comparison. If the current pixel analyzed deviates significantly from the background model, it is considered to be a part belonging to the foreground object and thus, it is labeled as a foreground pixel.

Depending on the algorithm used to separate the foreground from the background and the complexity of the image, the output will have different impacts in the noise quantity. Figure 2 shows an example of images involved in the foreground and background separation process[28].



**Figure 2. Background subtraction process[28], background model (left), original frame (center) and foreground extraction as a binary image (right)**

There are many different background algorithms designed to perform the background subtraction, each one with different advantages and limitations related to efficiency, quality, processing time and complexity. However, the algorithms have to fulfill certain requirements such as robustness: it must be adaptable to environmental changes, it must be fast enough to assure the information being analyzed is still meaningful and finally it has to consume as little computer resources as possible.

The most common techniques for background subtraction are described as follows[4]:

- *Frame Differencing*: This method is arguably the simplest form of background subtraction. The current frame is simply subtracted from the previous frame, and if the difference in pixel values for a given pixel is greater than a threshold  $T_s$ , the pixel is considered part of the foreground.

$$FG = |Frame_{(i)} - Frame_{(i-1)}| > T_s \quad (1)$$

This method does have two major advantages. One obvious advantage is the modest computational load. Another is that the background model is highly adaptive. Since the background is based solely on the previous frame, it can adapt to changes in the background faster than any other method. On the other hand, the method has serious flaws. The objects in the scene must be moving all the time, otherwise they will be recognized as background in subsequent frames.

- *Temporal Median Filter*: The information from previous frames is accumulated in order to get the average value for each pixel[5]. The median method creates a buffer of the last N frames and models the background as the median of those frames. This approach has proven to be very robust and have good performance in most applications. It is also very adaptative as the frame difference approach (although not as fast to adapt). However, this approach consumes a lot of memory resources as it is necessary to store several frames.
- *Approximate Median Method*: the previous N frames of video are buffered, and the background is calculated as the median of buffered frames. Then (as with frame difference), the background is subtracted from the current frame and thresholded to determine the foreground pixels. Median filtering has been shown to be very robust and to have performance comparable to higher complexity methods. However, storing and processing many frames of video (as is often

required to track slower moving objects) requires an often prohibitively large amount of memory. This can be alleviated somewhat by storing and processing frames at a rate lower than the frame rate, thereby lowering storage and computation requirements at the expense of a slower adapting background.

The approximate median method works as follows: if a pixel in the current frame has a value larger than the corresponding background pixel, the background pixel is incremented by one. Likewise, if the current pixel is less than the background pixel, the background is decremented by one. In this way, the background eventually converges to an estimate where half the input pixels are greater than the background, and half are less than the background—approximately the median (convergence time will vary based on frame rate and amount movement in the scene). This method is a very good compromise. It offers performance near what we can achieve with higher-complexity methods, and it costs not much more in computation and storage than frame differencing.

- *Running Gaussian Average*: In this approach, a Gaussian probability density function (PDF) is compared to the last N frames of a video sequence, and the average for each pixel is updated according to its previous values[4]. It is a mixture between the Gaussian and the median approximation.
- *Mixture of Gaussians (MoG)*: In this method, the background model is parametric instead of being a frame of values. Each pixel location is represented by a

number (or mixture) of Gaussian functions that sum together to form a probability distribution function  $F$  of the form:

$$F(i_t = \mu) = \sum_{i=1}^k \omega_{i,t} \cdot \eta(\mu, \sigma) \quad (2)$$

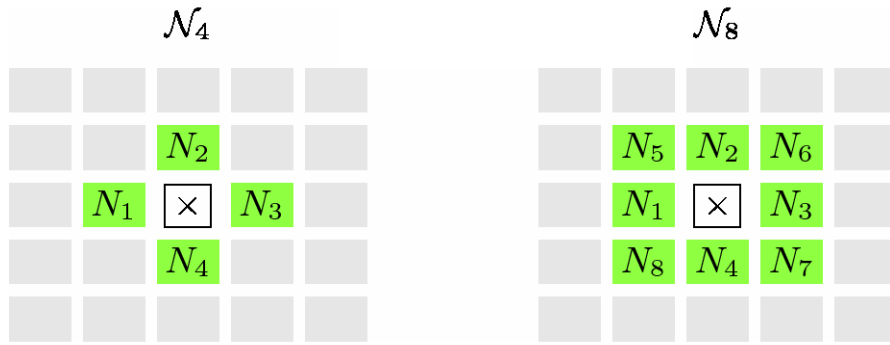
Where  $\mu$  corresponds to the mean of each Gaussian function and can be thought of as an educated guess of the pixel value in the next frame assuming that pixels are usually background. The weight ( $\omega$ ) and standard deviations ( $\sigma$ ) of each component are measures of the confidence in that guess (higher weight ( $\omega$ ) and lower standard deviation ( $\sigma$ ) lead to a higher confidence). There are usually 3 to 5 Gaussian components per pixel, the number typically depending on memory limitations. To determine if a pixel is part of the background, we compare it to the Gaussian components tracking it. If the pixel value is within a scaling factor of a background component's standard deviation ( $\sigma$ ), it is considered part of the background, otherwise it is foreground.

Other techniques used for background subtraction are: *Kernel density estimation (KDE)*, *Co-occurrence of image variations* and *Eigenbackgrounds*.

### 2.3.2 Mathematical Morphology

Mathematical morphology (MM) is a theory and technique for the analysis and processing of geometrical structures, based on set theory, lattice theory, topology, and

random functions. The basic idea in binary morphology is to probe an image with a simple, pre-defined shape, drawing conclusions on how this shape fits or misses the shapes in the image. This simple "probe" is called structuring element, and is itself a binary image which is usually composed by four neighboring pixels although there is a version composed by eight of them (Figure 3).



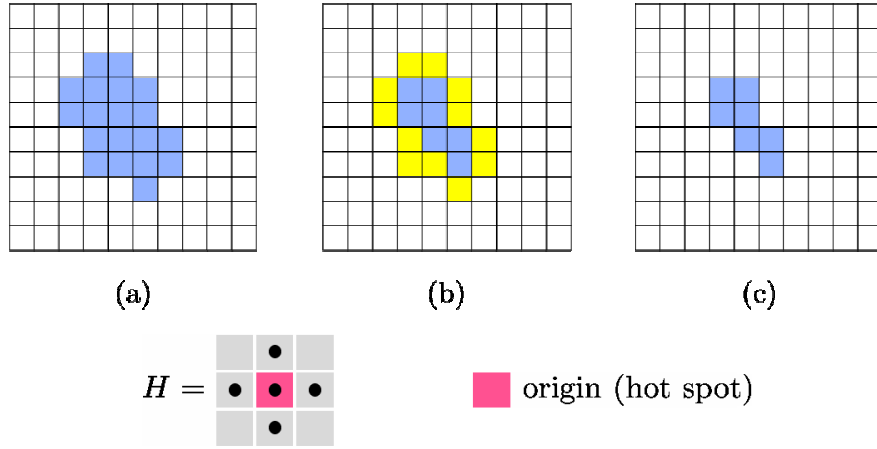
**Figure 3: Structuring element models**

Groups of neighboring grid centers (pixels) can be viewed as sets, and Boolean algebra can be applied to them. Morphological operations are then Boolean algebraic operations applied to the mapping of selected regions in a digital image. They can perform tasks such as finding the skeleton of a figure, filtering and restoration. To apply such methods the digital image has to be a binary image where the meaningful sectors are labeled one and the rest labeled zero.

There are four basic operations as building blocks of many operations in morphological image processing: Erosion, Dilation, Opening and Closing. These operations can be used to perform: Object extraction, image filtering operations (such as removal of small

objects or noise from an image), image segmentation operations (such as separating connected objects) and measurement operations (such as texture analysis and shape description).

### 2.3.2.1 Erosion



**Figure 4: Erosion example. (a) Original pixels, (b) Yellow pixels will disappear using the structuring element  $H$ , (c) resultant pixels**

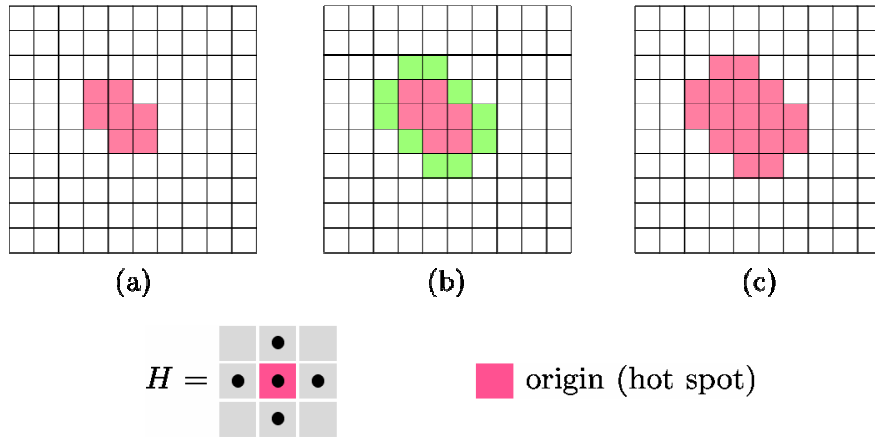
This operation shrinks the objects present in a binary image. The extent to which the objects thin depends on a controlling object referred to as the *structuring element*.

Mathematically, erosion is defined as:

$$A \ominus B = \{z \in E | B_z \subseteq A\} \quad (3)$$

This means that the erosion of image A by the structuring element B is the set of all structuring element locations that are not overlapping with the background of A in the entire grid system E.

### 2.3.2.2 Dilation



**Figure 5: Dilation example. (a) Original pixels, (b) Green pixels will appear using the structuring element H, (c) resultant pixels**

This operation enlarges the objects present in a binary image. The extent to which the objects grow depends on a controlling object referred to as the *structuring element*. Mathematically, dilation is defined as:

$$A \oplus B = \{z \in E | (B^s)_z \cap A \neq \emptyset\} \quad (4)$$

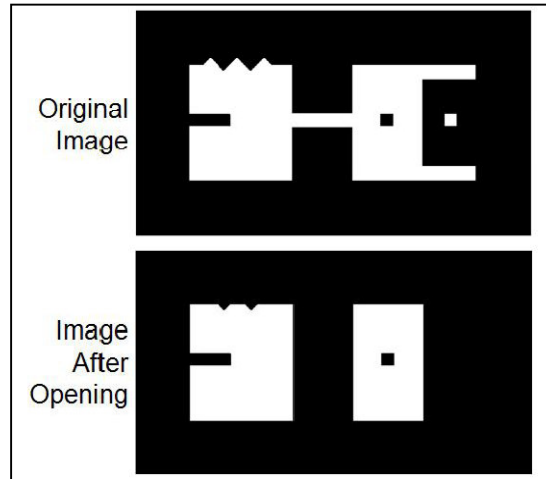
This means that the dilation of image A by the structuring element B is the set of all structuring element locations where the symmetric of B overlaps with at least a portion of A in the entire grid system E.



### 2.3.2.3 Opening

This operation is simply the erosion of  $A$  by a structuring element  $B$  followed by a dilation of the output by the same structuring element. In other words, opening is the Union of all possible locations of the structuring element  $B$  where  $B$  fits entirely inside  $A$ .

$$A \circ B = (A \ominus B) \oplus B \quad (5)$$



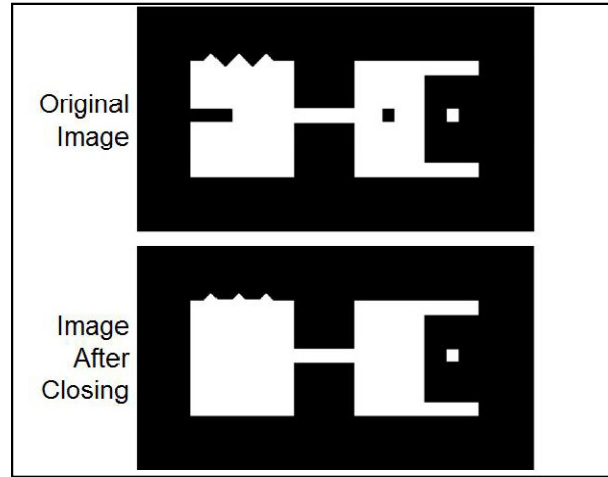
**Figure 6: Opening example using a 20-pixel square as structuring element[29]**

### 2.3.2.4 Closing

This operation is simply the dilation of  $A$  by a structuring element  $B$  followed by an erosion of the output by the same structuring element. In synthesis, closing is the

complement of the union of all possible locations of structuring element  $B$  where  $B$  fits entirely outside  $A$ .

$$A \bullet B = (A \oplus B) \ominus B \quad (6)$$

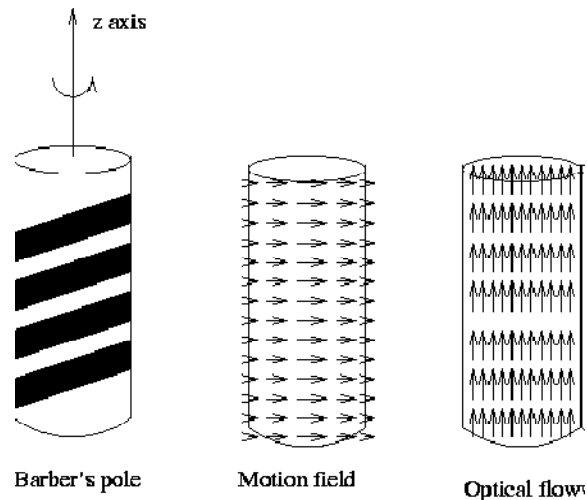


**Figure 7: Closing example using a 20-pixel square as structuring element[29]**

### 2.3.3 Optical Flow

Horn and Schunk defined the optical flow as the distribution of apparent velocities of movement of brightness patterns in an image[6]. We have to consider that optical flow determines velocities instead of displacements used in motion estimation; consequently it gives relevant information about the spatial arrangement of the objects viewed and the rate of change in this arrangement. Finally, optical flow can be seen as a velocity field which warps one image into another.

However, there is no strong relationship between the optical flow in the image plane and the velocities of the object when using more than two degrees of freedom world. An example of this can be seen in Figure 8 showing a barbershop pole [8] rotating along the  $z$  axis. The visual effect generated is horizontal lines going up and the optical flow detects in the barber's pole the brightness changes going from the bottom to the top. Therefore, we have to take special care about a given optical flow field; the information could be the result of brightness changes instead of real object motion. If an object is moving but there is no brightness change, optical flow is not able to identify rotation or translation.



**Figure 8: Optical flow and motion field of a barber's pole**

Some of the issues which could be interpreted as brightness change information instead of real object motion are: Brightness variation due to shading effects, Incident illumination and occluding edges. It is very important to consider the issues mentioned for further optical flow implementations.

### 2.3.3.1 The Optical Flow equation

The optical flow equation is based on the assumption that the defined pattern moving along the scene has the same brightness, so brightness pixels may change the original position to a new one while conserving the same intensity. Lets define  $x$  and  $y$  as the variables in a Cartesian coordinate system and  $t$  as the time variable, the brightness variable  $E$  of a particular point in the pattern is constant so:

$$\frac{d}{dt} E(x, y, t) = 0 \quad (7)$$

Equation (7) is the basis of Optical Flow: the intensity of a pattern along time and space remains constant. Using the chain rule for differentiation we obtain:

$$\frac{\partial E}{\partial x} \frac{dx}{dt} + \frac{\partial E}{\partial y} \frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \quad (8)$$

Replacing  $u = \frac{dx}{dt}$  and  $v = \frac{dy}{dt}$  we obtain:

$$E_x u + E_y v + E_t = 0 \quad (9)$$

Equation (9) shows a single linear equation with two unknown variables:  $u$  and  $v$ .  $E_x$ ,  $E_y$  and  $E_t$  correspond to the partial derivatives of image brightness with respect to  $x$ ,  $y$  and  $t$ , respectively. The previous equation can be rewritten as follows:

$$(E_x, E_y) \cdot (u, v) = -E_t \quad (10)$$

Thus the component of the movement in the direction of the brightness gradient  $(E_x, E_y)$  equals:

$$- \frac{E_t}{\sqrt{E_x^2 + E_y^2}} \quad (11)$$

Based on the previous analysis we can conclude:

- Equation (11) is not sufficient to uniquely specify the two dimensional velocity  $(u, v)$  because the single linear equation has two unknown variables.
- Equation (11) reveals that we can only estimate the component of the flow vector that is in the direction of the spatial image gradient. We cannot determine the component of the movement in the direction of iso-brightness contour; this is known as the “aperture problem”.

This point is where various optical flow techniques and algorithms were born. The following is a brief explanation of two of the most representative methods to evaluate the optical flow: The Horn-Schunck method and the Lucas-Kanade method.

### 2.3.3.2 Horn-Schunck method

This method includes a global constraint of smoothness to solve the aperture problem seeking a motion field that satisfies the optical flow equation with the minimum variation among the flow vector  $\mathcal{E}_b$  as follows:

$$\mathcal{E}_b = E_x u + E_y v + E_t \quad (12)$$

The optical flow equation is satisfied when  $\mathcal{E}_b=0$  but in practice, the image brightness measurements are corrupted by the quantization error and noise so we cannot expect  $\mathcal{E}_b=0$ . Then the method minimizes a weighted sum of the error in the optical flow equation and a measure of the pixel-to-pixel variation of the velocity field.

Gauss-Jordan elimination is a classical option to solve the minimization equations but is very expensive. The Gauss-Seidel method[8] is an iterative option which reduces the number of computational operations.

### 2.3.3.3 Lucas-Kanade method

This method divides the original image into smaller sections and assumes a constant velocity in each of these sections to solve  $u$  and  $v$  from the optical flow equation[7]. Assuming that  $u$  and  $v$  are constant in a small window of size  $m \times m$  with  $m > 1$  and numbering the pixels as 1, 2, ...,  $n$ , a set of equations can be found:

$$\begin{aligned} E_{x1}u + E_{y1}v &= -E_{t1} \\ \vdots \\ E_{xn}u + E_{yn}v &= -E_{tn} \end{aligned} \quad (13)$$

To solve the over-determined system of equations, the least squares method is used:

$$\begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_y I_x & \sum W^2 I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum W^2 I_x I_t \\ \sum W^2 I_y I_t \end{bmatrix} \quad (14)$$

$u$  and  $v$  can be solved by matrix inversion. Here,  $W$  is a window function that emphasizes the constraints at the center of each section.

### 2.3.4 Tracking

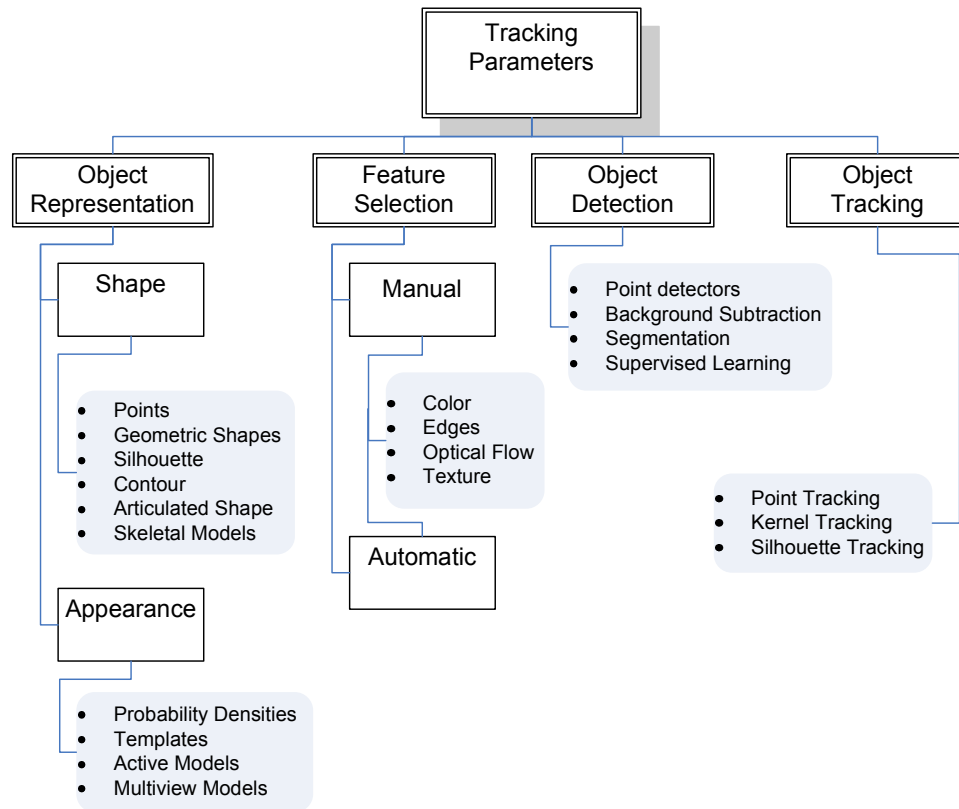
Different approaches have been explored to analyze the problem of tracking objects across multiple frames. The classification depends on the way the approach combines the following, bottom-up categorized, set of parameters [9,10]:

- *Object representation:* Objects can be represented according to their shapes (such as points, primitive geometric shapes, silhouette and contour, articulated shape and skeletal models) and appearances (such as probability densities, templates and active and multiview models).
- *Feature selection for tracking:* The most important factor is to distinguish the object from the others so, the uniqueness condition is the most desirable property of a visual feature. The usual visual features used for a tracking algorithm are: color, edges, optical flow and texture. Typically, features are selected manually, but in recent years interest in automatic feature selection has grown in the pattern recognition community. One of the most commonly used features is color; however, combination of different features is used to improve tracking performance.
- *Object detection:* Tracking methods need mechanisms to determine the presence of new objects in the scene. Some approaches use information from a single frame while others use accumulated information from a sequence of frames to improve robustness. Some of the methods that fall under this category are point detectors, background subtraction, segmentation and supervised learning.
- *Object Tracking:* Correlation among the different instances of an object along the frames that compose a video sequence results in object's trajectory. Tracking



methods are classified into three main categories: point, kernel and silhouette tracking.

Figure 9 depicts the taxonomy of the previously described tracking parameters[28].



**Figure 9: Taxonomy of tracking parameters[28]**

## **Chapter 3 PROPOSED SOLUTION**

### **3.1 Introduction**

This chapter contains the information related to the proposed solution to detect events in a video surveillance system. A detailed description of the blocks used to implement the system will be given, as well as the organization of the blocks in the proposed pipeline with the role and interaction among them will be explained in the following sections.

### **3.2 General Description**

The proposed solution is designed following the video surveillance guidelines reviewed in Chapter 2. Although the solution can be adapted to work in a real time environment in the future, it is focused to work with small video clips due to actual constraints in memory and computational resources. The video clips are extracted for research purposes using the event annotations given by the TRECVid annotators. The system processes the video frames to extract the information needed to accomplish the event detection purpose. It also analyses the information and finally gives an indication about the state of the event during the video playback.

### 3.2.1 Event Probability Zone Definition

There can be a big variety of events to detect in surveillance videos. The number of events depends on different factors which the system must consider according to the design parameters, purpose of detections, camera locations and probability of events occurring in specific locations.

There are ten events defined for TRECVID 2009[25]: *PersonRuns*, *CellToEar*, *ObjectPut*, *PeopleMeet*, *PeopleSplitUp*, *Embrace*, *Pointing*, *ElevatorNoEntry*, *OpposingFlow* and *TakePicture*. Each event has to be analyzed according to its definition and description in order to be modeled in the system. This thesis is focused in two specific events as explained in Section 1.3: *OpposingFlow* and *PersonRuns*.

**Table 1: Event descriptions**

EVENT	DESCRIPTION	START TIME	END TIME
PersonRuns	Someone runs	The earliest time the subject is visibly running	The latest time the subject is visibly running
OpposingFlow	Someone moves through a door opposite to the normal flow of traffic	The earliest time when the person has begun to move through the door	When the person has fully passed through the doorway

The initial point to model the system architecture is based on the event probability concept. There are certain zones in the video which have a higher probability of events occurring compared than others do. The two events selected for detection are related to the motion of objects, so it is natural to analyze the motion in specific areas where movements are expected and discard the remaining ones. Figure 10 shows an image with

an example of area selection to detect motion for running events; the predominant area where events are detected is located in the sidewalks next to the pool. Motion is not expected in the pool area, the sky or the buildings, so these zones are discarded.

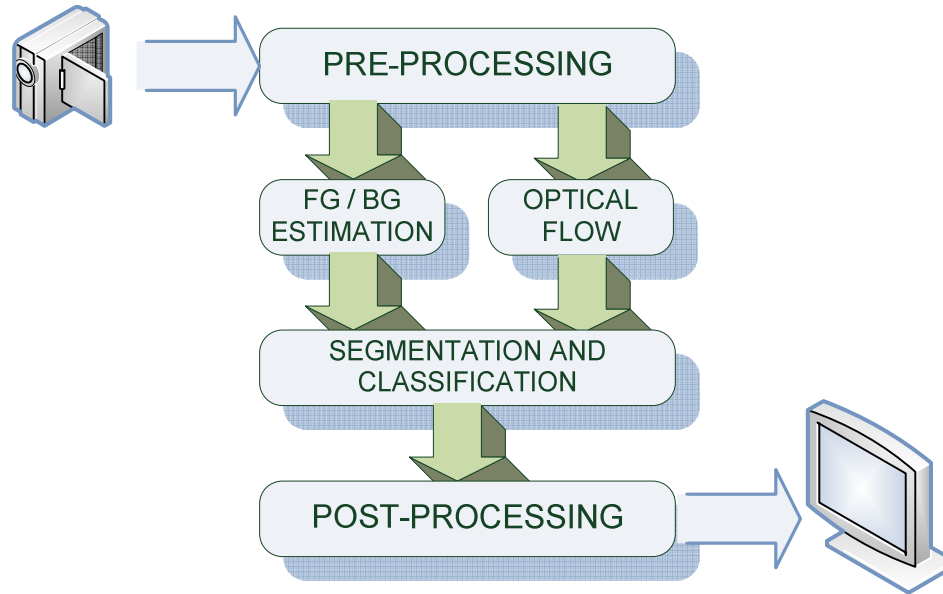


**Figure 10: Event probability model. Original frame (left), selected zones for event detection (right)**

### 3.2.2 Block Diagram

The general block diagram for the proposed solution is represented in Figure 11, which shows the system composed of five blocks. The input video is pre-processed to decode it in frames and determine the selection of relevant information which will make the event detection task easier to achieve. The Foreground/Background Estimation block and the Optical Flow block receive the pre-processed information. The first separates the background from the foreground; and the second evaluates the optical flow for all the pixels contained in the pre-processed image. After that, the optical flow information is selected and classified in the next block based on the foreground contents and the conditions proposed to flag the selected event. Finally, the data is post-processed to

determine whether or not the event is occurring, so that the user can visualize the results obtained.



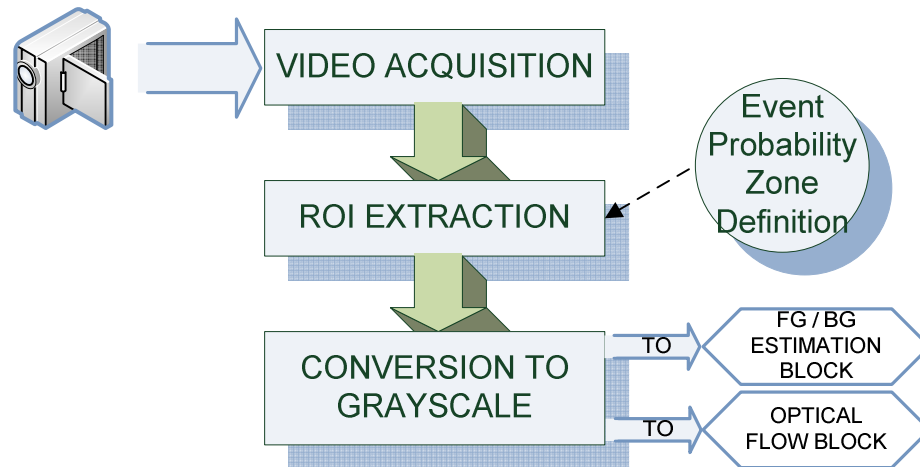
**Figure 11: General Block Diagram**

### **3.3 Detailed Description**

In the ensuing sub sections, a description of the proposed solution will be provided.

#### **3.3.1 Pre-processing**

The pre-processing block is composed of three different stages (Figure 12) where the input video is received to extract the region of importance in which the event is defined. It is then converted from the RGB color space to a set of grayscale images for further processing.



**Figure 12: Pre-processing block diagram**

### 3.3.1.1 Video Acquisition

In this stage, the video input is analyzed to extract relevant information, such as format, frames per second, height, width, number of frames, quality, compression, etc. Using the aforementioned information, the video is received to be decoded and decompressed in a structure array containing the RGB matrix representation of each frame in the video.

### 3.3.1.2 Region of Interest (ROI) Extraction

Due to the nature of the event being detected, it is highly recommended to determine specific areas where occurrences tend to take place in order to focus the detection analysis in the most suitable parts where the event existence probability is higher. This concept implies a deep analysis of the environmental conditions and objects behavior

given by the experimental observation of the video context and the specific event definition and conditions.

Using the region of interest in the event detection task will help to decrease the probability of false detections. It will be also useful to reduce the complexity of the system in future processing tasks because the decreased use of memory resources.

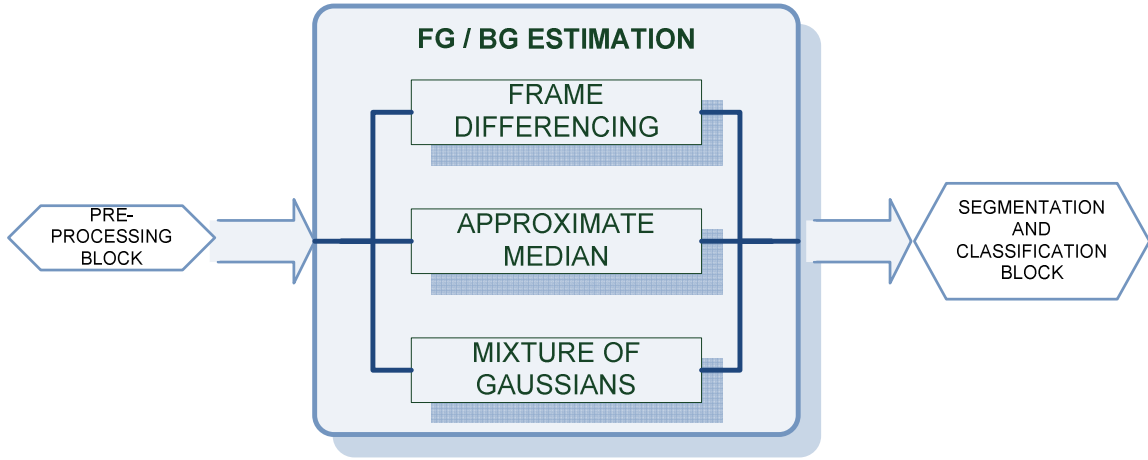
### 3.3.1.3 Conversion to Grayscale

Color components in images and video sequences can be useful when trying to process specific features' detection, such as skin color for human or face recognition. However, many approaches, such as the evaluation of the optical flow, require only grayscale images to perform the operations.

Grayscale images are related to the luminance component in the YCbCr color space, where Y is the luminance component that gives the average brightness of the image. Since working with the luminance component is enough to process images without chrominance components, conversion from the RGB color space to the YCbCr color space is done in order to work only with the Y component as a grayscale sequence of images.

Working only with one component instead of three, leads to a system to be computationally more efficient since less memory resources are used.

### 3.3.2 Foreground/Background Estimation



**Figure 13: FG/BG Estimation block diagram**

The Foreground/Background estimation block receives the grayscale information contained in the ROI to separate the Foreground from the Background for further analysis of information contained in the Foreground. The output consists of binary frames where zero (0) corresponds to background pixels and one (1) represents foreground pixels. Three different methods for Background subtraction are analyzed (Figure 13) in order to determine the best solution that complement the environmental modeling conditions at a reasonable and efficient computational cost.

#### 3.3.2.1 Frame Difference

This method is analyzed as a low complexity approach. It is fast and easy to implement. However, it is highly dependent on continuous motion and very sensitive to noise.



### 3.3.2.2 Approximate Median

This method is analyzed as a medium complexity approach. It is easy to implement and is more robust than the Frame Difference method. It offers performance near what we can achieve with higher-complexity methods, therefore is less sensitive to noise.

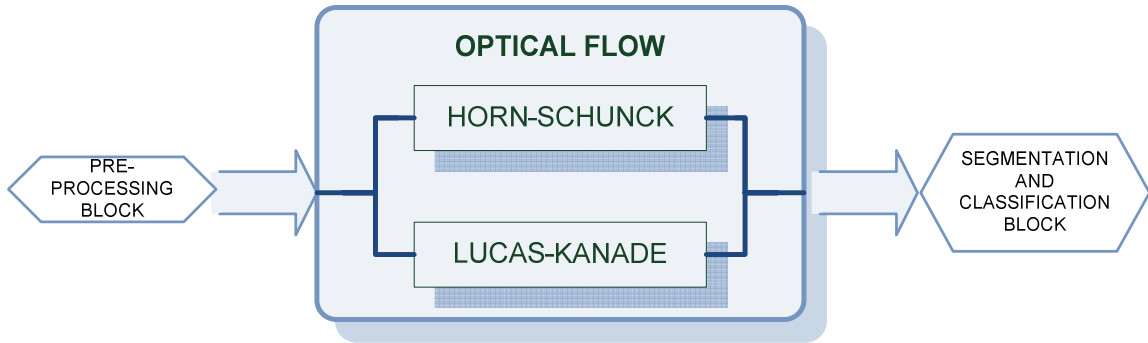
### 3.3.2.3 Mixture of Gaussians

This method is analyzed as a high complexity approach. Its implementation is more difficult because it has to deal with the parameter optimization of its five variables that generate significant impact on the performance of the algorithm. Although this method is more elegant, it consumes a significantly larger amount of computer resources and time than the former.

### 3.3.3 Optical Flow

The Optical Flow block also receives the grayscale information contained in the ROI to evaluate the optical flow for each pixel in every frame. The output is expressed as a complex number where the real part represents the optical flow value in the  $x$  axis and the imaginary part represents the optical flow value in the  $y$  axis. Two methods to evaluate

the optical flow are analyzed (Figure 14) in order to determine the most efficient computational cost solution.



**Figure 14: Optical Flow block diagram**

#### 3.3.3.1 Horn-Schunck method

The Horn-Schunck algorithm calculates the optical flow vector for each single pixel in an image. The algorithm has high accuracy, but has to be tuned using two parameters:

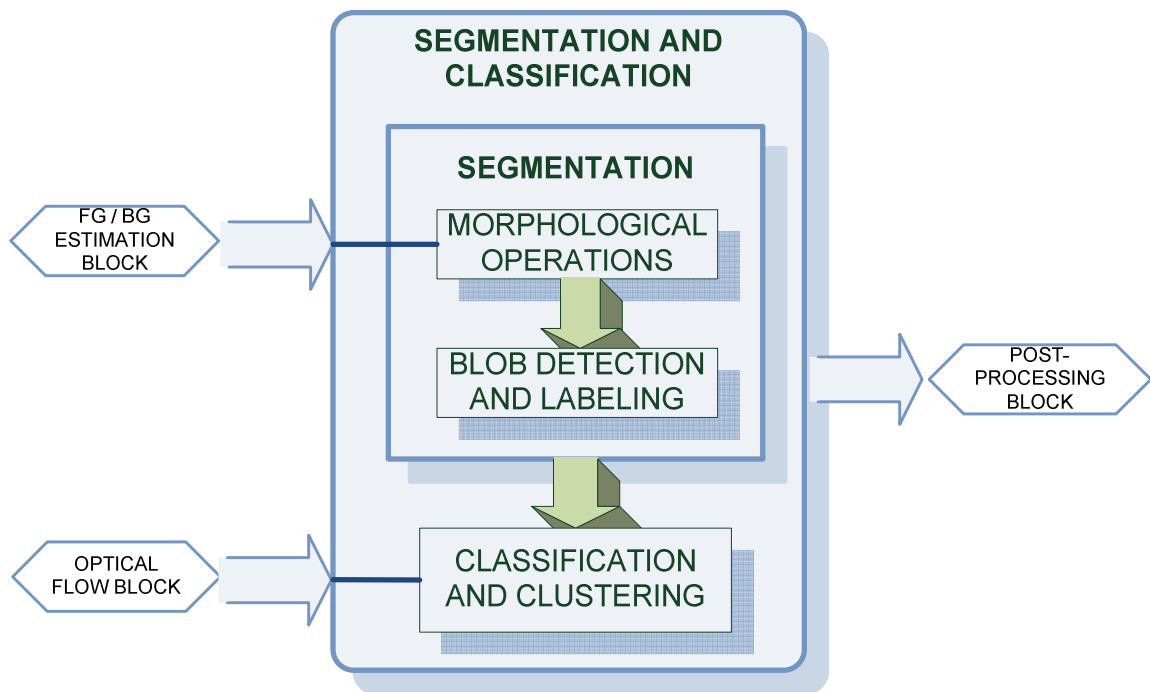
- Number of iterations.
- Weighting factor for the total error minimization.

Moreover, computation time is extremely lengthy because of its computational complexity.

### 3.3.3.2 Lucas-Kanade method

The Lucas-Kanade algorithm is based on the fact that the original image is divided into smaller sections assuming a constant velocity in each of these sections. The algorithm has acceptable quality, low computational complexity and works using only one input parameter: WindowSize. Greater values for the window size reduce the computation time.

### 3.3.4 Segmentation and Classification



**Figure 15: Segmentation and classification block diagram**

Using the information provided by the Foreground/Background estimation block, the segmentation and classification block (Figure 15) performs the segmentation of blobs identified in the foreground extracting relevant information such as area, centroid, number of blobs, etc.

Every detected and segmented blob will have its corresponding set of optical flow vectors per pixel belonging to the blob area. These optical flow vectors will be used to calculate a resultant optical flow vector to characterize each blob's dominant direction of movement.

According to the specific event definition, resultant optical flow vectors are identified, counted and clustered when they match the conditions to flag the event. For OpposingFlow event, optical flow vectors have to be inside the angle interval where the OpposingFlow event is defined. For PersonRuns event, optical flow vectors have to be greater than the threshold defined to make the difference between walking and running. The other resultant optical flow vectors are dismissed in both cases.

#### 3.3.4.1 Segmentation

Using morphological operations, the different areas in the foreground, which are given by the background subtraction algorithm, are processed to define consistent blobs and

suppress noise. Small detected areas are removed and the remaining areas are expanded using a structuring element; finally, the possible holes inside the blob are filled.

The blobs are then labeled to be properly identified and counted in every frame. Also, properties, such as area, centroid and bounding box, are measured for each blob for further processing purposes.

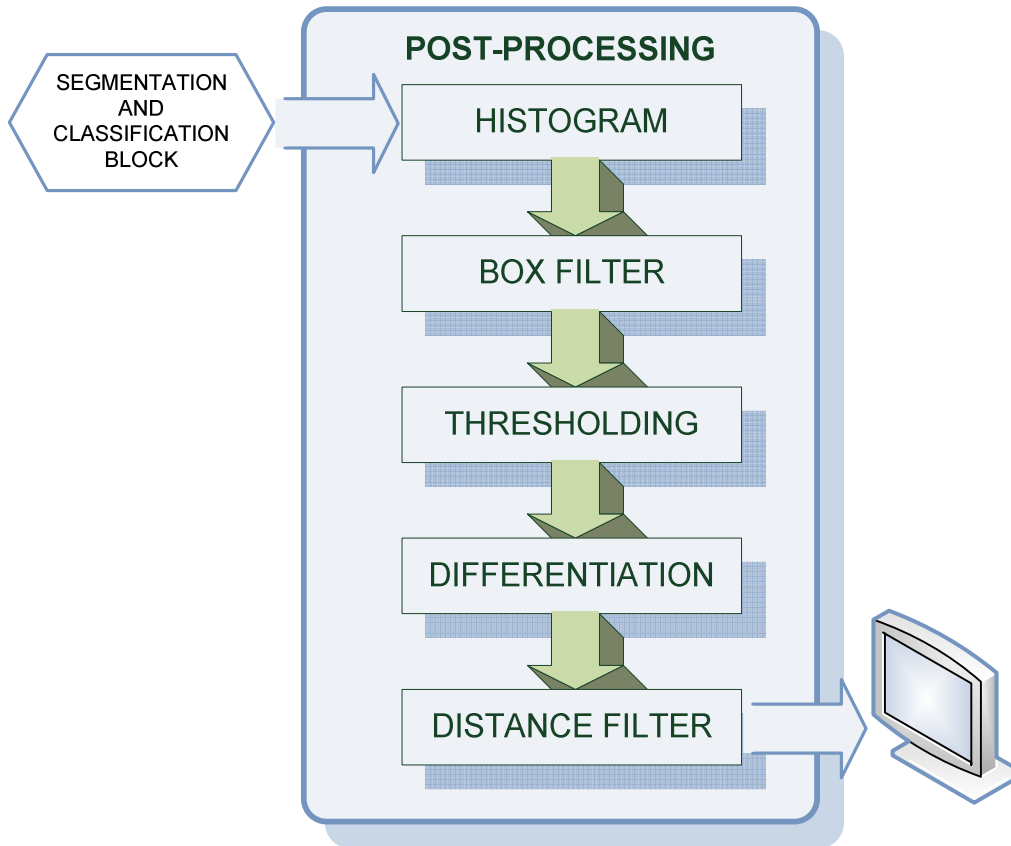
#### 3.3.4.2 Classification and Clustering

Every blob detected and labeled must be processed to determine the magnitude and dominant direction of its movement. This task is done using the blob's area as a mask to average all the optical flow vectors belonging to the blob; the resultant vector represents its predominant motion direction.

At this point, every blob has information that needs to be classified according to the event detection definition in order to resolve whether the blob is a candidate matching the criteria or not. To accomplish this goal, the phase value of the resultant optical flow vector is analyzed to find if it is inside the angle interval in which the OpposingFlow event is defined in the case of OpposingFlow event detection. In the case of PersonRuns event detection, the magnitude of the resultant optical flow vector is analyzed to find the values over a certain threshold showing higher motion activity which is a direct consequence of running events. Optical flow vectors outside these conditions are dropped

and the ones matching the conditions are clustered to be processed to determine the existence of the desired event.

### 3.3.5 Post-processing



**Figure 16: Post-Processing block diagram**

The last step in the system corresponds to the Post-processing block (Figure 16) which receives the clustered information from the previous block and creates a histogram with the number of optical flow vectors per frame. A convolution operation is performed

between this data and a small constant window to act as a low pass filter to increase the confidence in detection under the assumption that when the event is happening, it is supposed to last during several contiguous frames. Then, the convolution results are compared to a threshold, which is calculated empirically, to determine the candidate frames where the event is happening, creating a new histogram where values equal to one (1) correspond to candidate frames and values equal to zero (0) correspond to non-candidate frames.

Due to some possible discontinuities in the threshold stage, the presence of occlusions, noise and other factors in the classification of candidates, it may happen that the new histogram has non-continuous values for a detected event as well as isolated presence of candidate frames, which can make the final decision inaccurate leading to false event detections. To solve these issues, the new histogram is differentiated so we can extract the start frame and the end frame for the candidate events. The starting point of a detected event is identified with a positive one (+1) and the ending point of the event is identified with a negative one (-1).

The final decision is made by analyzing the distances among start-end points as well as end-start points. Distance between start points and end points is performed in order to discard detected events which fail to comply with the minimum duration required to tag the event. On the other hand, distance between end points and start points is intended to identify cases where the event should be continuous; but it has small discontinuities leading to tag different instances of the same event detected.

Finally, the system is able to specify the start frame and end frame where the event has been identified. This information is useful to perform some statistical analysis according to event annotations based on the ground truth. Moreover, the data is used to show the tagged event during playback, so as to act as an interface with the end user of the video surveillance system.



## **Chapter 4 IMPLEMENTATION**

### **4.1 Introduction**

The most relevant information about implementation aspects is included in this chapter. The software used and detailed explanations for the implementation of the different parts of the algorithm are also provided.

The algorithm implementation was done in MATLAB version 7.6.0 using the Image Processing Toolbox (IPT) version 6.1. The aforementioned IPT contains the main tools used to implement the algorithms described in this thesis.

The implementation consists in one main algorithm for each event, hosting all the different stages mentioned in Chapter 3. The algorithm is implemented as a function that receives the video input filename, the desired background subtraction method and the desired optical flow method as parameters. The output is an array which contains the information about the number of detected events and their duration. For statistical purposes, the algorithm returns an output describing the percentage of true and false detections.

## 4.2 Pre-Processing

### 4.2.1 Video Acquisition

A set of video clips were extracted from the dataset of videos provided by TRECVID. Every video clip was extracted using the TRECVID ground truth information to assure the existence of only one event. The ground truth specifies the initial frame and the final frame a specific event is taking place in the video, so the extraction was performed using some frames before and after the event. FFMPEG open source library was used to extract the video clips using the following command line:

```
ffmpeg.exe -i [Input File] -vcodec copy -an -f rawvideo -ss [Start Time] -t [Duration]  
[Output File]
```

MATLAB IPT uses the function *aviread* to load video files in AVI format. Other video formats must be converted to AVI in order to work with this function in the program. In this case, the video files were originally saved in MPEG format, so using the FFMPEG open source library, the video clips were converted to AVI format using the command line:

```
ffmpeg.exe -i [Input File] -sameq [AVI Output File]
```

The first step of video acquisition process is extracting the video information which is going to be useful to configure properly the algorithm. Information, such as Number of

frames, Frames per second, Height, Width, Quality, etc, is stored in a structure whose fields contain information about the AVI file specified in the parameter of the function *aviinfo* in the IPT. With the function *aviread* in the IPT, every frame in the AVI file is stored as an array of structures where each position in the array corresponds to the structure containing the RGB values for each frame expressed in a 3-D matrix with the dimensions: Height x Width x 3. The number of elements in the array corresponds to the number of frames in the video.

#### 4.2.2 Region of Interest (ROI) Extraction

According to the OpposingFlow event definition: “Someone moves through a door opposite to the normal flow of traffic”[25], the region of interest is restricted to cover the area of the doors in the video, specifically the top area, in order to avoid occlusions and undesired motion of people in the scene without crossing the doors area. For general purposes, MATLAB IPT has the function *roipoly* to define interactively the desired ROI.

#### 4.2.3 Conversion to Grayscale

Usually, the process to extract the luminance component from RGB color space implies the conversion to YCbCr color space to work only with the Y component.

Fortunately, MATLAB IPT has the function *rgb2gray* to extract automatically the grayscale image from the RGB color space. The new grayscale video is stored in a

variable *gray\_frames* which is a Matrix with dimensions: ROI-Height x ROI-Width x Number of frames.

### 4.3 Foreground/Background Estimation

According to the main function call, the system is able to perform three different methods to separate the foreground from the background. The variable containing the information (*bgs*) is compared to determine which method is going to be used. Every method is implemented as a function receiving the corresponding parameters and returning two variables, *FG* with the foreground information and *BG* with the background information. *FG* and *BG* are matrices with dimensions: ROI-Height x ROI-Width x Number of Frames.

The possible values for *bgs* are: FD (*Frame Differencing*), AM (*Approximate Median*) and MoG (*Mixture of Gaussians*). The default background subtraction implementation is based on the *Approximate Median Method*.

#### 4.3.1 Frame Difference

The function *bg\_frame\_diff* implements the frame difference method as explained in Section 2.3.1. The parameters in this function are: the grayscale video and the desired

threshold. The returned values are the foreground and background information explained in Section 4.3.

#### 4.3.2 Approximate Median

The function *bg\_approx\_media* implements the approximate median method as explained in Section 2.3.1. The parameters in this function are: the grayscale video and the desired threshold. The returned values are the foreground and background information explained in Section 4.3.

#### 4.3.3 Mixture of Gaussians

The function *bg\_mog* implements the Mixture of Gaussians method as explained in Section 2.3.1. The parameters in this function are: the grayscale video, the number of Gaussian components, the number of background components, the positive deviation threshold, the learning rate, the foreground threshold and the initial standard deviation. The returned values are the foreground and background information explained in Section 4.3.

## 4.4 Optical Flow

According to the main function call, the system is able to perform two different methods to evaluate the optical flow. The variable containing the information (*of*) is examined to determine which method is going to be used. Every method is implemented as a function receiving the corresponding parameters and returning two variables, *FlowInfo* with the optical flow components for every pixel and *FlowSize* with information related to the size of the previous variable. Additionally, those outputs are rearranged into a new variable to express the optical flow as a complex vector for every single pixel in the video frames. This new variable, called *OF\_complex*, is a matrix with dimensions: ROI-Height x ROI-Width x Number of Frames.

The possible values for *of* are: HS (*Horn-Schunck*) and LK (*Lucas-Kanade*). The default optical flow implementation is based on the *Lucas-Kanade Method*.

### 4.4.1 Horn-Schunck method

The function *HornSchunck*[30] implements the *Horn-Schunck method* as explained in Section 2.3.3.2. The parameters in this function are: the current frame, the previous frame, the weighting factor *alpha* and the number of iterations. The returned values are the optical flow components for every pixel as explained in Section 4.4

#### 4.4.2 Lucas-Kanade method

The function *LucasKanade*[30] implements the *Lucas-Kanade method* as explained in Section 2.3.3.3. The parameters in this function are: the current frame, the previous frame and the window size. The returned values are the optical flow components for every pixel as explained in Section 4.4

### 4.5 Segmentation and Classification

#### 4.5.1 Segmentation

##### 4.5.1.1 Morphological Operations

A series of morphological operations are performed for each of the foreground images in order to remove isolated pixels, connect adjacent points and remove smaller objects. MATLAB IPT functions are used to perform these operations; first removing small objects, then adjacent points are connected using a disk as structuring element and finally, possible holes are filled to assure consistency in the objects.

##### 4.5.1.2 Blob Detection and Labeling

Every detected object is labeled after the morphological operation using another function in MATLAB IPT. There will be a number of objects detected for each frame; each one

with a label and a set of characteristics such as Area, BoundingBox and Centroid extracted by the MATLAB IPT function *regionprops*. This information will be used to mask the Optical Flow vectors belonging to each detected object.

#### 4.5.2 Classification and Clustering

Every object detected in the foreground is used as a mask to compute the average of its Optical Flow vectors in order to extract a single resultant vector for each object. This vector is going to represent the magnitude and direction for every object detected in the frame. A new vector of structures is going to be used to store the resultant optical flow vector for each object. This structure is also going to store all the single optical flow vectors belonging to the object and the respective object's centroid for visualization purposes.

According to the event definition for OpposingFlow, the resultant optical flow vectors with phase greater or equal to 90 and less or equal to 180 must be selected as candidates for the event detection. This condition in phase is equivalent to have negative values in the horizontal optical flow component and positive values in the vertical optical flow component.

In the case of PersonRuns event, the resultant optical flow vectors with magnitude greater than one must be selected as candidates for the event detection.



It is very important to notice that the coordinate (0,0) in the image is located in the upper left corner so vertical movements are inverted in the polar representation of the vectors.

Finally, the number of candidates for the desired event detection are counted and clustered for each frame for further analysis.

## **4.6 Post-Processing**

### **4.6.1 Histogram**

The contents in the variable *candidates* are the values used to generate the histogram of candidates per frame matching the specified criteria. When the event is happening, it is expected to have greater number of candidates during an interval of time. When the event is not happening, the candidates should be zero, but depending on the environment conditions such automatic doors opening and closing, noise, occlusions and human gestures, there can be few candidates so it is necessary to filter the values on the histogram to increase the event detection confidence.

### **4.6.2 Box Filter**

A convolution operation is performed with the information in the histogram and a small window with fixed size to generate the variable *prealarm*. MATLAB function *conv* was

used for this purpose. The window size was selected based on observation of the motion and the speed factor of the elements moving in the selected area.

#### 4.6.3 Thresholding

A threshold value is assigned to the variable  $k$  to set the level of comparison respect to the variable *prealarm*. Values greater than  $k$  will be stored as ones in a new variable called *th\_prealarm*, otherwise they will be stored as zeros. The value for the variable  $k$  was selected empirically based on observations according to the number of objects moving during the presence of the event in the video.

#### 4.6.4 Differentiation

Depending on the variance of the variable *prealarm*, it may be possible to have discontinuities after the thresholding operation leading to have multiple detections of the same event. MATLAB function *diff* was used to differentiate the *th\_prealarm* variable in order to determine the start-end points of the detected events. Thus, the starting point of the event will be identified with a positive one (+1) and the ending point of the event will be identified with a negative one (-1). The rest of the points will be zero (0). The new variable used to store these values is called *toggle*.

#### 4.6.5 Distance Filter

This filter is intended to analyze the distance between start-end points as well as end-start points.

First, the analysis is done between the end point of a detected event and the start point of the next detected event. For every detected event, if the distance between the end point and the next start point is less than a minimum predefined value, the points are removed. This action removes discontinuities in the detection of the event to prevent multiple detections of the same event.

Finally, the analysis is done between start-end points, which is equivalent to analyze the duration of the event. For every detected event, if the distance between the start point and the end point is less than a minimum predefined value, the points are removed. This action can be interpreted as the elimination of short event detections.

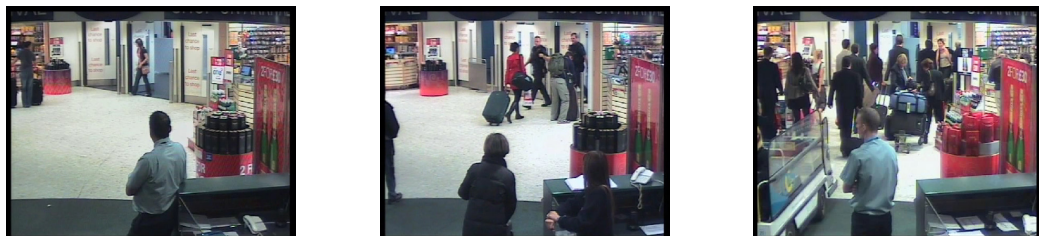
The predefined value is calculated based on the minimum number of frames to mark the presence of the event in the video and it is stored in the variable *min\_dist*.

The final output corresponding to the definitive event detection is stored in the variable *event* which is an array with dimensions: 2 x Number of Detected Events. For every detected event, the start frame is stored in the first row, and the end frame is stored in the last row.

## Chapter 5 EXPERIMENTS AND RESULTS

### 5.1 Introduction

Results of experiments are provided in this chapter according to the implementation explained in Chapter 4. A set of eleven videos divided in three different groups were analyzed to tune the system for event detection. Each group was determined according to the video conditions and complexity to detect events such as crowded environments and occlusions (Figure 17). Group one contains five videos without a crowded environment and low probability of occlusions, group two contains three videos with some occlusions, and group three contains three videos fully crowded leading to a higher probability of occlusions.



**Figure 17: Groups of Videos. Group 1 (left), Group 2 (center) and Group 3 (right)**

The computer used to test the program was a Dell Optiplex 745 with the following specifications:

- Intel Core 2 Duo Processor GX620 @ 2.4GHz
- DDR2 SDRAM, 4096MB @ 667MHz
- NVIDIA GeForce 8800 GT Display Adapter, 512MB
- Windows XP SP2, MATLAB 7.6.0 (R2008a), IPT Version 6.1

Processing time was measured for every block described in Chapter 3 focusing on the Foreground/Background Estimation block and the Optical Flow block in order to analyze the viability to make a real-time implementation in the future.

Finally, a statistical analysis of performance and detections will be found at the end of the chapter.

## **5.2 Pre-Processing**

### **5.2.1 Video Acquisition**

The video clips used to test the program were extracted from the video database provided by TRECVID, where OpposingFlow and PersonRuns events have the same environmental conditions (Camera1). Every clip was extracted according to the ground-truth of detected

events assuring the existence of only one event per clip. Additionally, every clip contains 100 frames before the event starts and another 100 frames after the event finishes (Table 2 and Table 3).

**Table 2: Set of videos and length for OpposingFlow event detection**

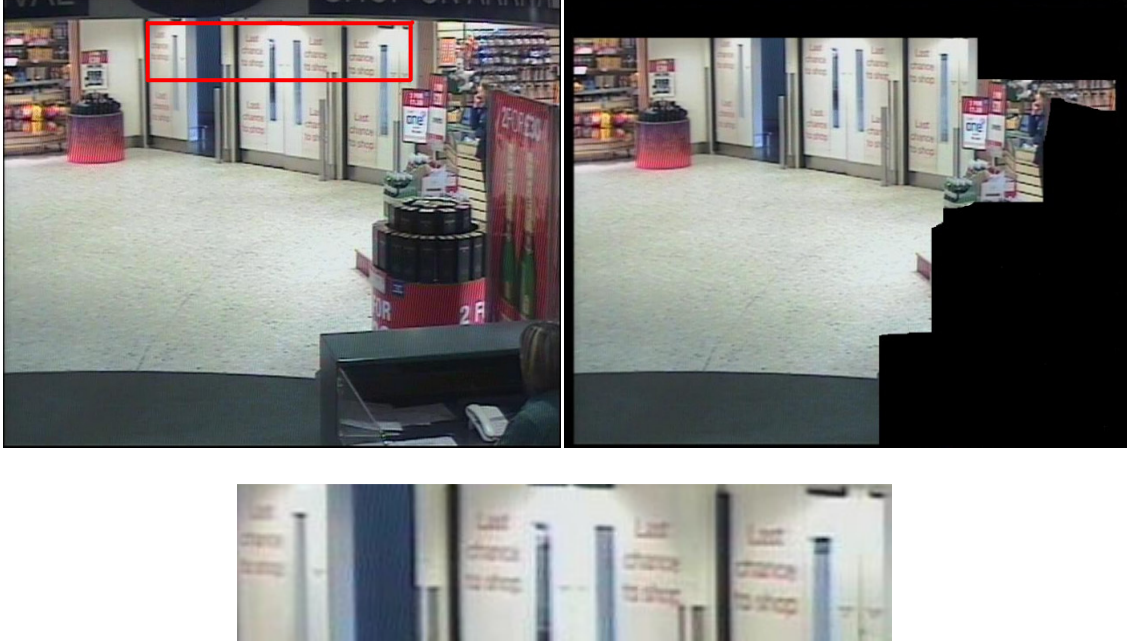
<b>VIDEO</b>	<b>NUMBER OF FRAMES</b>
Video_01	251
Video_02	226
Video_03	226
Video_04	251
Video_05	226
Video_06	226
Video_07	326
Video_08	226
Video_09	251
Video_10	301
Video_11	426
Video_12	226
Video_13	301
Video_14	226
Video_15	301
Video_16	201
Video_17	226
Video_18	251
Video_19	226
Video_20	226
Video_21	351
Video_22	426
Video_23	226
Video_24	226
Video_25	226
Video_26	226
Video_27	226
Video_28	226
Video_29	251
Video_30	251
Video_31	226
Video_32	251
Video_33	226

**Table 3: Set of videos and length for PersonRuns event detection**

<b>VIDEO</b>	<b>NUMBER OF FRAMES</b>
Video_01	251
Video_02	276
Video_03	301
Video_04	251
Video_05	251
Video_06	251
Video_07	326
Video_08	226
Video_09	301
Video_10	301
Video_12	276
Video_13	226
Video_14	251
Video_15	251
Video_16	251
Video_18	301
Video_19	226
Video_20	251
Video_21	226
Video_22	326
Video_23	251
Video_25	276

### 5.2.2 Region of Interest (ROI) Extraction

The region of interest was extracted according to the area with the highest probability of the event to occur. Although the area of the OpposingFlow event is defined to happen in the doors zone, only the top area of the doors was selected in order to avoid occlusions and undesired motion of people walking from the right to the left in the scene without crossing the doors, but has a limitation because it won't let the system to detect kids or small people crossing the doors. In the case of PersonRuns event, areas with furniture and objects, as well as areas close to the ceiling were discarded due to the reduced probability of events to occur.



**Figure 18: ROI selection and extraction. Frame from original video (top-left), ROI for PersonRuns (top-right), extracted ROI for OpposingFlow (bottom)**

Figure 18 shows the regions selected from the original video and the extracted image. This approach was also useful to decrease computational loads when processing the Background/Foreground Estimation and the Optical Flow. For example, the original video resolution was reduced to 15.81% in the case of OpposingFlow event detection after the ROI extraction.

### 5.2.3 Conversion to Grayscale

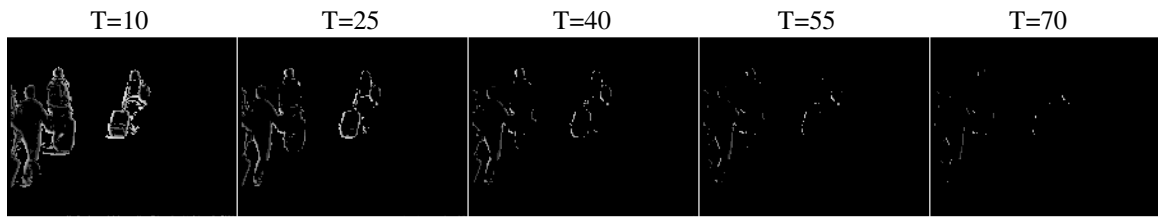
The conversion to grayscale was also useful to reduce the computational load to 1/3. Instead of processing three color components, the process is made using only the luminance component.



## 5.3 Foreground/Background Estimation

### 5.3.1 Frame Difference

The frame difference algorithm has the threshold parameter which was changed for different values (threshold=10, 25, 40, 55 and 70) according to the observations showed in Figure 19. Higher values reduce the existence of foreground pixels while lower values lead to the presence of noise; 25 was the most suitable value to reduce the noise and have enough foreground pixels.



**Figure 19: Threshold comparison for Frame Difference method**

### 5.3.2 Approximate Median

The approximate median algorithm also has the threshold parameter which was changed for different values (threshold=10, 25, 40, 55 and 70) according to the observations showed in Figure 20. Higher values reduce the existence of foreground pixels while lower values lead to the presence of noise, 25 was the most suitable value to reduce the noise and have enough foreground pixels.

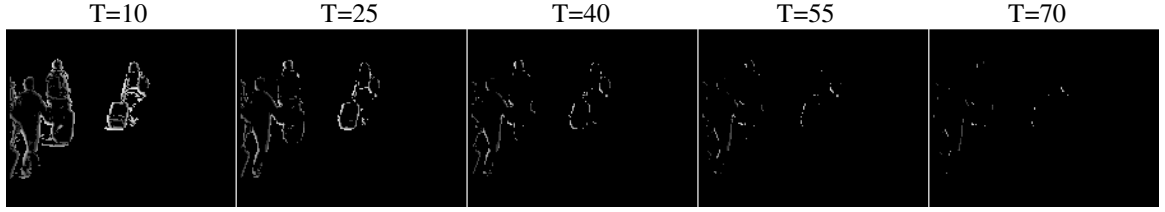


Figure 20: Threshold comparison for Approximate Median method

### 5.3.3 Mixture of Gaussians

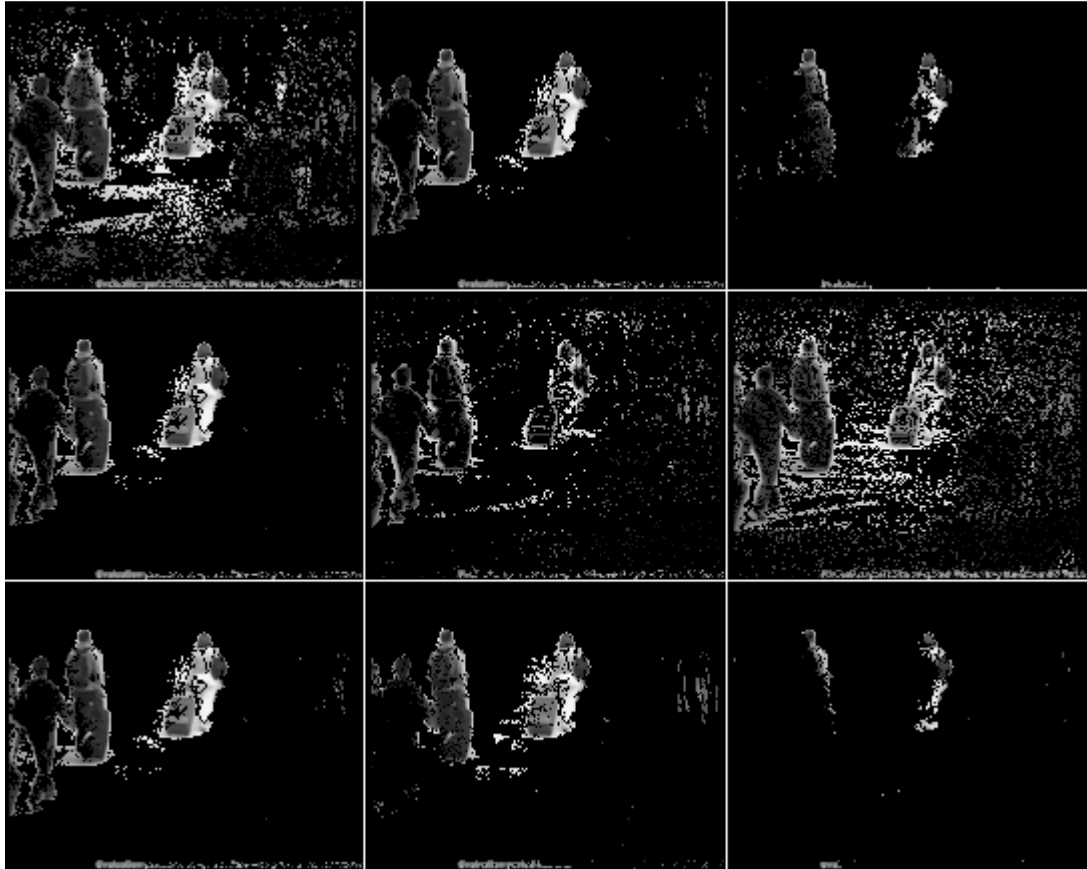


Figure 21: Parameter test for Mixture of Gaussians method.

The mixture of Gaussians algorithm has more than one parameter to change which makes the tuning process cumbersome due to the combination of values to find the best configuration. Three parameters were evaluated ( $\text{dev\_thresh}=1, 2.5$  and  $5$ ,  $\alpha=0.01$ ,

0.1 and 0.5 and finally, thresh=0.25, 0.5 and 0.75) to determine the most suitable combination of values to reduce the noise and have enough accurate pixels in the foreground. The approach was to test one parameter at a time: while sweeping one parameter, the others remain fixed to see how the algorithm performs. Then another parameter is swept, and finally the values with the best results after combinations were chosen. Figure 21 shows some of the images according to the parameter test: the deviation threshold is swept in the first row, the learning rate in the second row and the foreground threshold in the last row.

## 5.4 Optical Flow



Figure 22: Horn-Schunk optical flow vectors.

The optical flow methods used to evaluate the optical flow vectors of pixels in every different frame were successfully tested. The Horn-Schunck method showed an accurate evaluation of vectors, but the processing time is very long as it is shown in the time analysis section. Figure 22 shows a very dense optical flow field in the zoomed image because there is one single optical flow vector per pixel.



**Figure 23: Lucas-Kanade optical flow vectors.**

On the other hand, the Lucas-Kanade method performs much faster in processing time and the optical flow density can be handled changing the WindowSize parameter as it is shown in Figure 23 where the WindowSize value is set to three. We can see the distance between every optical flow vector is equivalent to the WindowSize value; however, this parameter has to be carefully selected because the classification task explained in Section 4.5.2 performs better with higher density of optical flow vectors.

## 5.5 Segmentation and Classification

### 5.5.1 Segmentation

The segmentation process shown in Figure 24 performs the morphological operations to create the blobs based on the foreground information. First, the *opening* operation was used to remove small objects, then the *dilation* operation was used using a disk as a structuring element and finally, the *closing* operation was used to fill regions and holes; this process was able to clean part of the noise and expanded the edges of the objects detected on the foreground.

The shapes after the segmentation process result in blobs that are far from ideal because they are not good representation of semantic objects. However, every detected blob was labeled as a different object in the frame and represented with a distinguishable color.

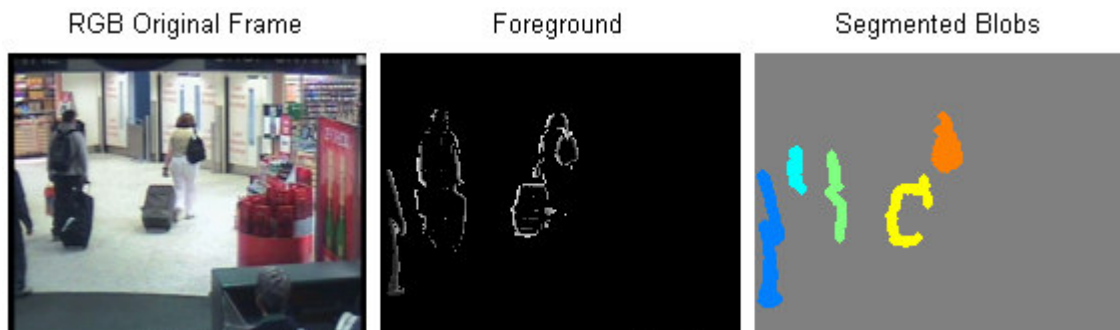
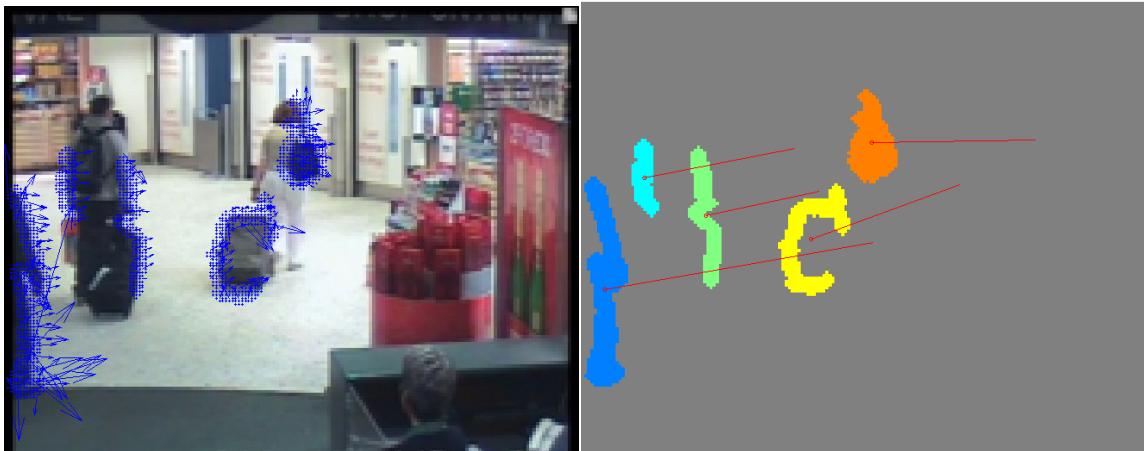


Figure 24: Segmentation process

### 5.5.2 Classification and Clustering

Every labeled blob has a number of optical flow vectors according to its area. For each blob, a resultant optical flow vector is calculated discarding the optical flow vectors whose value is zero and measuring the average of the remaining components; the resultant vector is plotted in the centroid of the blob so we can have an overall idea of the magnitude and direction of movement of the objects in the scene.

Figure 25 shows the original image with the optical flow vectors belonging to every detected blob and the resultant vector located in the centroid of the blobs.

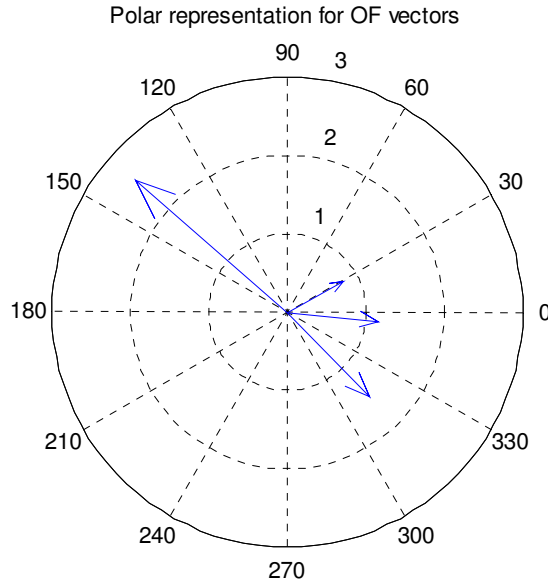


**Figure 25: Optical flow vectors after segmentation. Vectors belonging to the blobs (left) and resultant vectors per blob (right).**

The resultant optical flow vectors are also plotted in a polar coordinate system to analyze the magnitude and angle of displacement (Figure 26). When the magnitude is greater



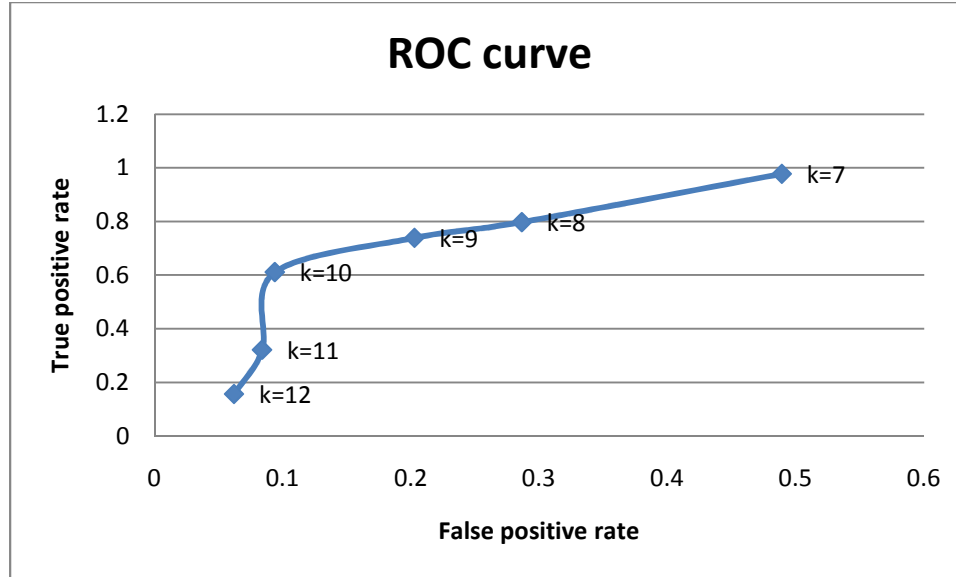
than one, the vector is a candidate for PersonRuns events, and when the angle is between 90 and 180 degrees, the vector is a candidate for OpposingFlow events.



**Figure 26: Optical flow vectors plot.**

## 5.6 Post-Processing

According to the event detection classification, every vector matching the desired criteria is counted to generate the histogram of candidates per each frame. The histogram is filtered performing a convolution operation with a fixed size window to act as a low pass filter and the resultant values are thresholded to determine the most suitable candidates for the final decision. The window size for the convolution was set at fifteen (15), and the threshold value was changed from 7 to 12 determining the best threshold value to have a reasonable percentage in true detections.



**Figure 27: Threshold values to optimize the detections percentage**

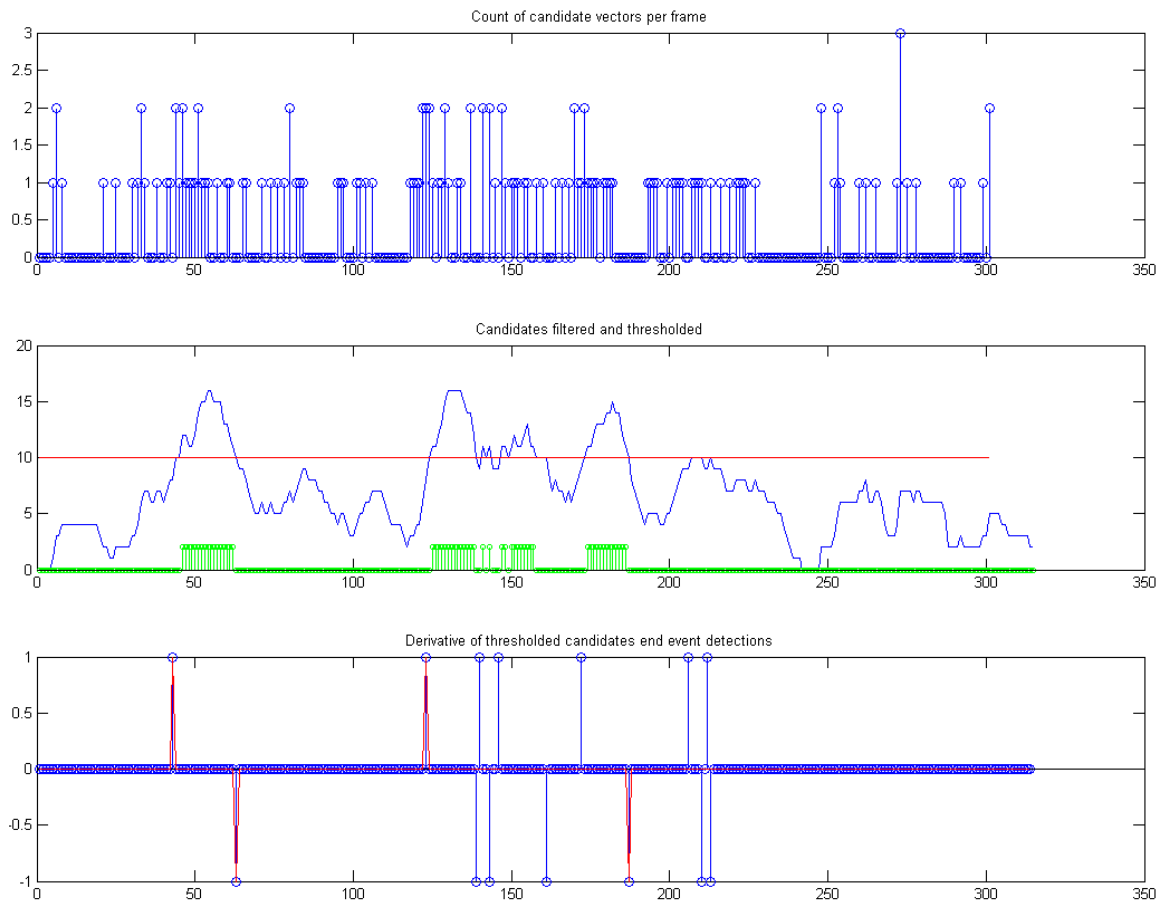
Figure 27 shows the percentage in detections according to the different threshold values used in the set of videos for OpposingFlow event detection. When the threshold value increases, the false detections tends to decrease, and the true detections tend to increase; finally the threshold value was set to 10 as the most suitable value.

After the values are thresholded, they are differentiated to find the start and end points for every detected event in the video clip to perform a comparison with the ground truth annotations. A positive one (+1) corresponds to a start point and a negative one (-1) corresponds to an end point. However, due to the variance in detections, occlusions and other factors, there can be discontinuities in the detection of one single event and detections that need to be discarded because the extreme short duration. A minimum distance value is set to 16 based on the assumption that a real event should last at least 16 consecutive frames; events shorter than the minimum distance are discarded and events



separated by a distance less or equal to the minimum distance specified are joint together. The final detection corresponds to the start and end points after the distance analysis.

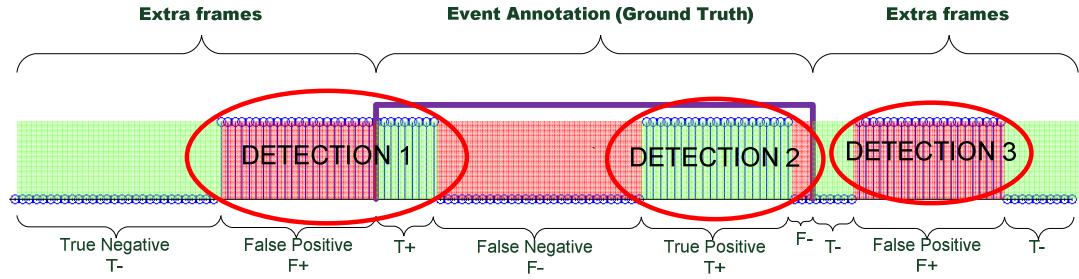
Figure 28 shows the different steps of post-processing: the first plot corresponds to the histogram, the second plot corresponds to the filtered signal (blue), the threshold value (red) and the thresholded values (green). Finally the last plot shows the different start and end points of the thresholded events (blue) as well as the final detections after the distance filter applied to the differentiation process (red).



**Figure 28: Post-processing stage. Histogram of candidates (top), candidates filtered and thresholded (middle), identification and filtering of start and end points for final detected events.**

## 5.7 Statistics

The classification of detections (true and false detections) is calculated based on the information given by the ground truth. As every video clip was extracted in order to contain only one single event, and the video clips have extra frames before and after the event, it is expected to have the true detections during the window when the event is happening according to the ground truth annotation. Therefore, the false detections are going to take place when the detection is outside the event annotation, in other words, when there is a detection that belongs to the extra frames in the video.



**Figure 29: Detection statistics based on ground truth annotations**

Figure 29 shows an example where the system performed three different detections. Detection 1 is identified as a false detection, although it has some frames inside the event annotation window, most of the frames are outside. Detection 2 is identified as a true detection because it is entirely inside the event annotation window. Finally, detection 3 is identified as a false detection because it is completely outside the event annotation

window, in other words, it belongs to the extra frames in the video where there are not expected detections.

### 5.7.1 Opposing Flow event

Statistics for OpposingFlow event were evaluated using the different background subtraction techniques and optical flow implementations. The values obtained demonstrate that the different combinations of techniques perform satisfactorily, and also confirm that according the processing time demanded by every technique, the preferred ones are the approximate median method for background subtraction and the Lucas-Kanade method for optical flow implementation. Table 4 shows the average time every block spent to process the set of videos for OpposingFlow event detection.

**Table 4: Time analysis for OpposingFlow events (Values given in seconds)**

<b>Optical Flow</b>	<b>Background subtraction</b>	<b>Metric</b>	<b>Pre-process</b>	<b>FG/BG</b>	<b>OF</b>	<b>Segment and Classif</b>	<b>Post-process</b>	<b>Total</b>
<b>HS</b>	FD	Time	7.60	0.38	2563.46	10.21	0.01	2581.67
		Profile	0.29%	0.01%	99.29%	0.40%	0.00%	100%
	AM	Time	7.60	0.54	2563.46	10.21	0.01	2581.83
		Profile	0.29%	0.02%	99.29%	0.40%	0.00%	100%
	MoG	Time	7.60	145.64	2563.46	10.21	0.01	2726.93
		Profile	0.28%	5.34%	94.01%	0.37%	0.00%	100%
<b>LK</b>	FD	Time	7.60	0.38	66.32	10.21	0.01	84.53
		Profile	8.99%	0.45%	78.46%	12.08%	0.02%	100%
	AM	Time	7.60	0.54	66.32	10.21	0.01	84.69
		Profile	8.98%	0.64%	78.31%	12.06%	0.02%	100%
	MoG	Time	7.60	145.64	66.32	10.21	0.01	229.79
		Profile	3.31%	63.38%	28.86%	4.44%	0.01%	100%

Considering the length of videos in Table 2, the estimated average time to process one frame at a time can be seen in Table 5. Note that the processing times using the Mixture of Gaussians and the Horn-Schunck approach are extremely high for a real time implementation.

**Table 5: Time analysis to process a frame in OpposingFlow events (Values given in seconds)**

Optical Flow	Background subtraction	Metric	Pre-process	FG/BG	OF	Segment and Classif	Post-process	Total
<b>HS</b>	FD	Time	0.03	0.00	10.48	0.04	0.00	10.55
		Profile	0.29%	0.01%	99.29%	0.40%	0.00%	100%
	AM	Time	0.03	2.24E-03	10.48	0.04	0.00	10.55
		Profile	0.29%	0.02%	99.28%	0.40%	0.00%	100%
	MoG	Time	0.03	0.60	10.48	0.04	0.00	11.15
		Profile	0.28%	5.34%	94.00%	0.38%	0.00%	100%
<b>LK</b>	FD	Time	0.03	0.00	0.27	0.04	0.00	0.35
		Profile	8.97%	0.45%	78.39%	12.17%	0.02%	100%
	AM	Time	0.03	2.24E-03	0.27	0.04	0.00	0.35
		Profile	8.95%	0.65%	78.24%	12.15%	0.02%	100%
	MoG	Time	0.03	0.60	0.27	0.04	0.00	0.94
		Profile	3.31%	63.31%	28.89%	4.49%	0.01%	100%

Finally, the number of detections can be seen in Table 6. Results show that the number of true detections is acceptable for event detection having a recall of 81.8% and precision of 64.28%. It is very important to mention that these values depend on the method used to analyze the performance of the system (video clips with only one event) and they will decrease substantially when adapting the system to process an entire video with several frames, as the original videos provided by TREC Vid, because there is a high probability to have many more false detections in longer videos. The aforementioned information is supported by a test performed in a different set of 60 video clips, extracted randomly

from the TRECVID video dataset where each video clip had 301 frames; the result was 26 false detections.

**Table 6: Detections for OpposingFlow events**

VIDEO	DETECTIONS	
	FALSE	TRUE
Video_01	1	1
Video_02	0	1
Video_03	0	0
Video_04	0	1
Video_05	1	1
Video_06	0	1
Video_07	1	1
Video_08	0	1
Video_09	1	1
Video_10	1	1
Video_11	1	1

VIDEO	DETECTIONS	
	FALSE	TRUE
Video_12	0	1
Video_13	1	1
Video_14	0	1
Video_15	2	1
Video_16	0	0
Video_17	0	0
Video_18	0	1
Video_19	0	1
Video_20	0	1
Video_21	0	1
Video_22	1	1

VIDEO	DETECTIONS	
	FALSE	TRUE
Video_23	1	1
Video_24	0	0
Video_25	1	1
Video_26	1	1
Video_27	0	0
Video_28	0	0
Video_29	1	1
Video_30	0	1
Video_31	0	1
Video_32	0	1
Video_33	1	1
<b>TOTAL</b>	<b>15</b>	<b>27</b>

Recall and precision (Figure 30 and Figure 31) are evaluated using different values for the threshold  $k$  and the minimum distance  $min\_dist$  when performing the filtering stage. It is noticeable the recall is dependent on the threshold and the precision is dependent on the minimum distance. As the value for the threshold is incremented, the recall tends to decrease because the system is dropping detections below the threshold. On the other hand, as the minimum distance increases, the precision tends to increase but there is not a strong dependence with the value used for thresholding.

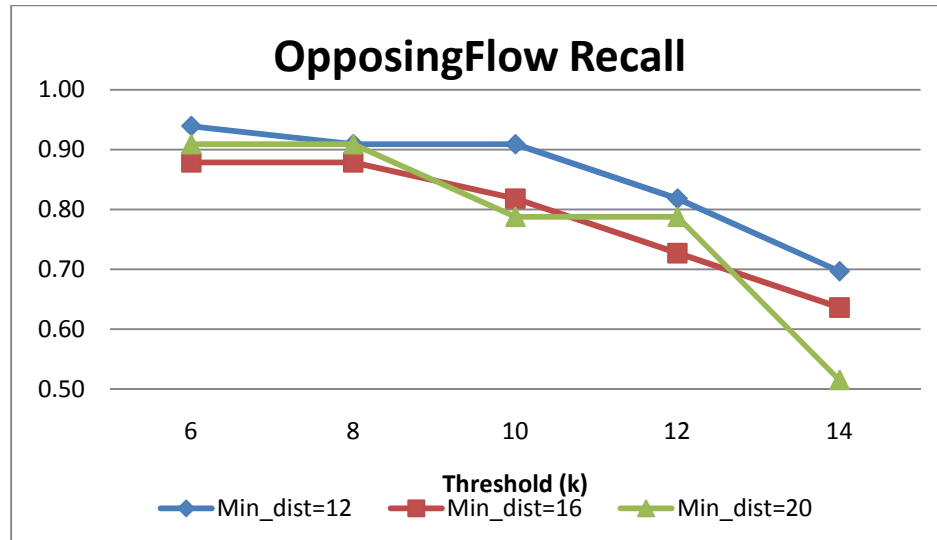


Figure 30: OpposingFlow Recall

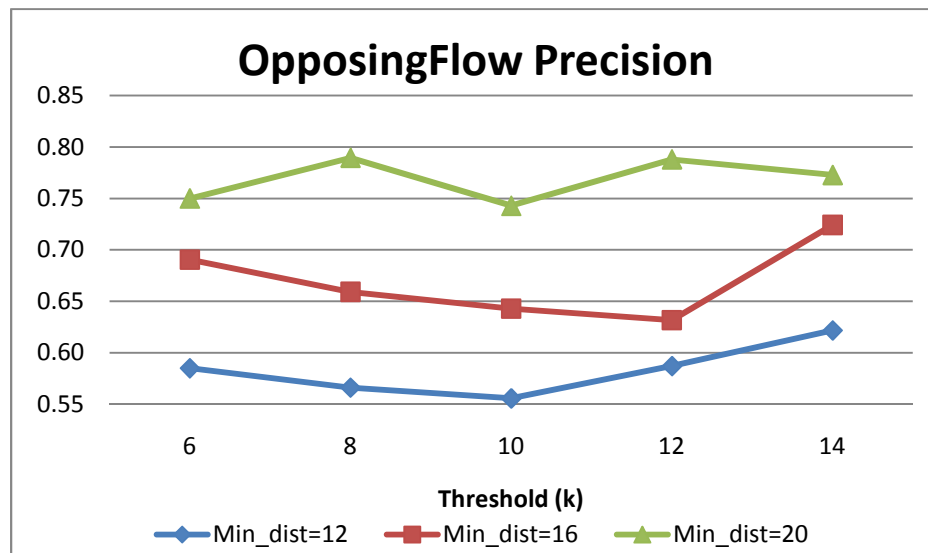


Figure 31: OpposingFlow Precision

Figure 32 shows the ROC curve for OpposingFlow detections. The minimum distance has a minimal impact in the different curves, but it is noticeable that higher values for the true positive rate make increase the false positive rate.

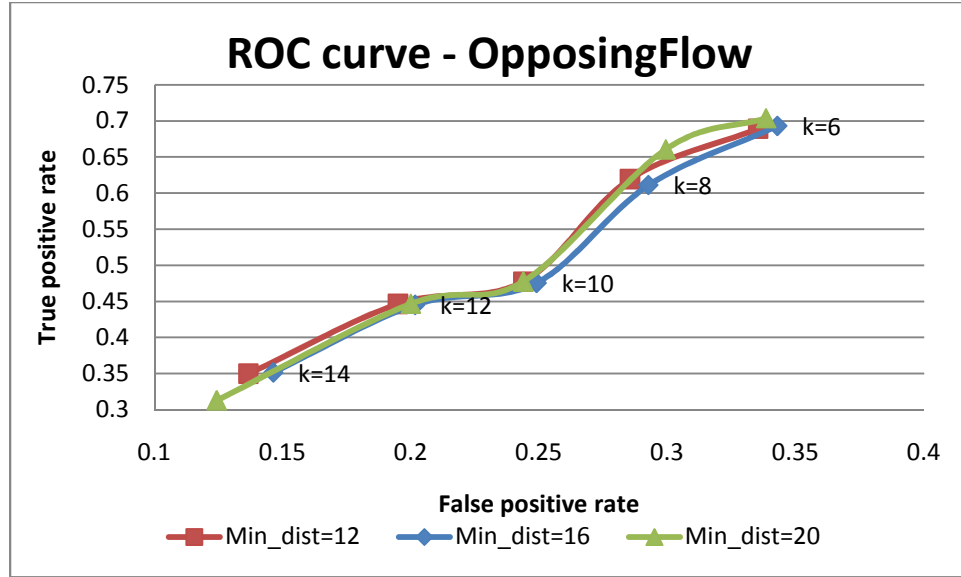


Figure 32: Threshold impact for OpposingFlow

### 5.7.2 Person Runs event

Statistics for PersonRuns events were evaluated using the approximate median method for background estimation and the Lucas-Kanade algorithm for optical flow implementation.

Table 7: Time analysis for PersonRuns events (Values given in seconds)

Metric	Pre-process	FG/BG AM	OF LK	Segmentation and Classification	Post-process	Total
Time	14.537	0.548	306.819	9.298	0.0053	331.209
Profile	4.39%	0.17%	92.64%	2.81%	0.00%	100%

Table 7 shows the average time every block spent to process the set of videos for PersonRuns event detection. It is noticeable that processing times are greater compared

with OpposingFlow events because the different amount of information according to the selected ROI.

Considering the length of videos in Table 3, the estimated average time to process one frame at a time can be seen in Table 8.

**Table 8: Time analysis to process a frame in PersonRuns events (Values given in seconds)**

<b>Metric</b>	<b>Pre-process</b>	<b>FG/BG AM</b>	<b>OF LK</b>	<b>Segmentation and Classification</b>	<b>Post-process</b>	<b>Total</b>
<b>Time</b>	0.057	0.00215	1.241178	0.0367071	2E-05	1.33707
<b>Profile</b>	4.26%	0.16%	92.83%	2.75%	0.00%	100%

Finally, the percentage of detections can be seen in Table 9. Results show that the percentages of true detections are acceptable for event detections having a recall of 81.8% and precision of 50%, even though the overall performance is lower compared to the OpposingFlow events.

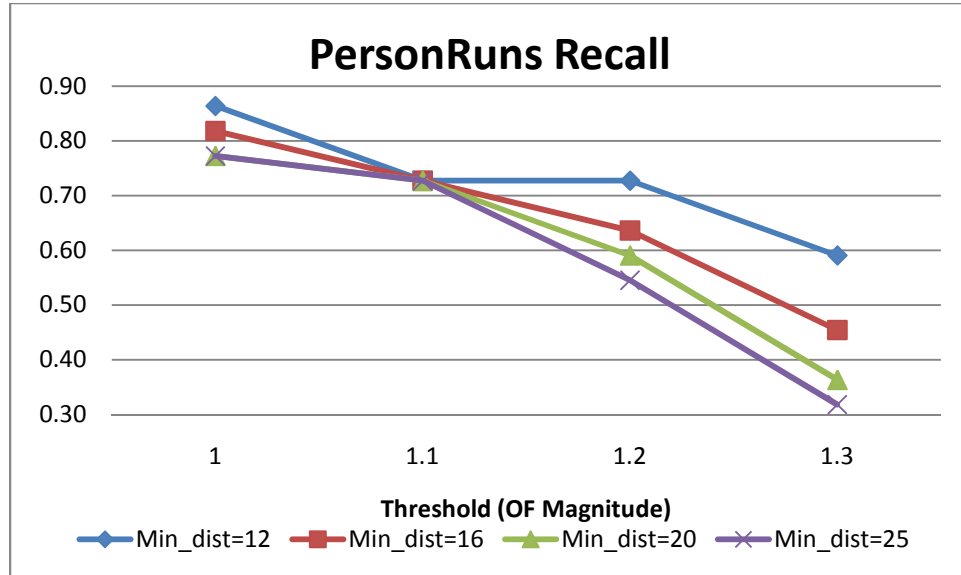
It is very important to mention that these values depend on the method used to analyze the performance of the system (video clips with only one event) and they will decrease substantially when adapting the system to process an entire video with several frames, as the original videos provided by TRECVid, because there is a high probability to have many more false detections in longer videos.



**Table 9: Detections for PersonRuns events**

VIDEO	DETECTIONS		VIDEO	DETECTIONS	
	FALSE	TRUE		FALSE	TRUE
Video_01	0	1	Video_13	2	1
Video_02	0	1	Video_14	1	1
Video_03	0	1	Video_15	2	1
Video_04	1	0	Video_16	2	0
Video_05	2	1	Video_18	0	1
Video_06	1	1	Video_19	0	1
Video_07	1	1	Video_20	1	1
Video_08	0	0	Video_21	1	1
Video_09	0	1	Video_22	1	1
Video_10	0	1	Video_23	1	0
Video_12	1	1	Video_25	1	1
<b>TOTAL</b>			<b>18</b>	<b>18</b>	

Recall and precision (Figure 33 and Figure 34) are evaluated using different values for the threshold *magnitude* used when evaluating the resultant optical flow vector, and the minimum distance *min\_dist* used when performing the filtering stage.



**Figure 33: PersonRuns Recall**

Although the precision is low, its value is better for higher values assigned to the minimum distance but the cost is a reduction in the recall. On the other hand, lower values for the threshold reflect a better recall. As the threshold value increases, the recall goes down because there are fewer objects identified as running.

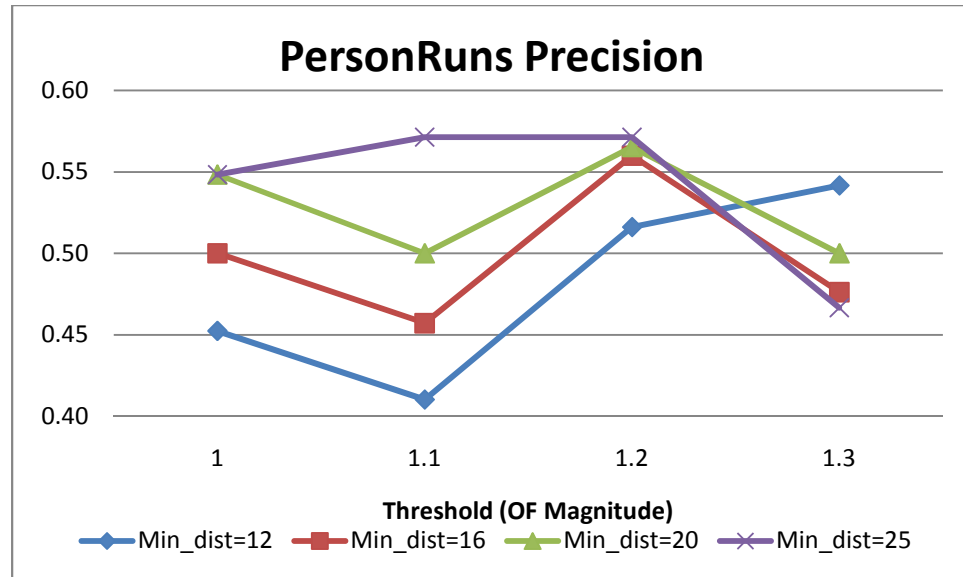


Figure 34: PersonRuns Precision

Figure 35 shows the ROC curve for PersonRuns detections. It is very clear how lower values for threshold lead to a higher true positive rates, but the tradeoff is the increment in the false positive rates.

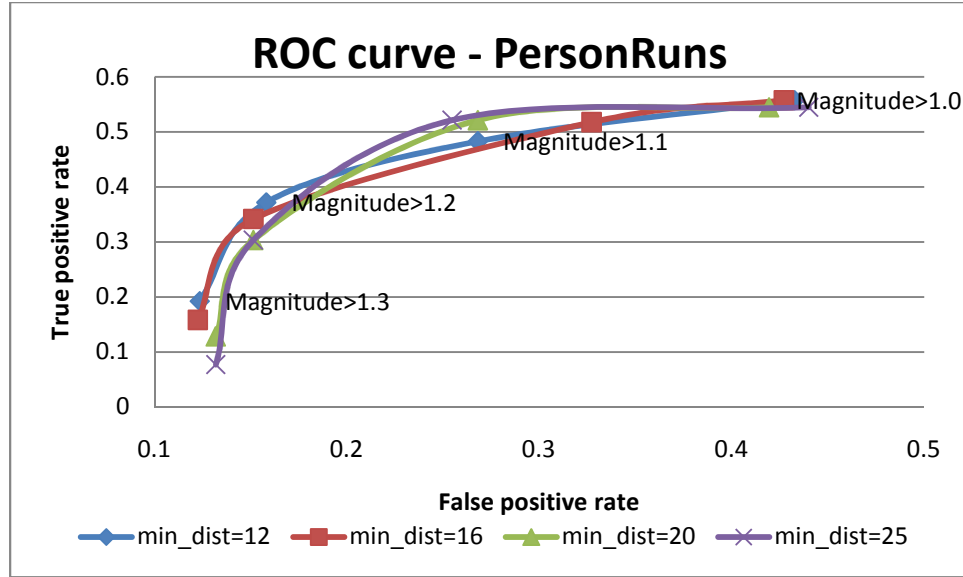


Figure 35: Threshold impact for PersonRuns

## 5.8 Summary

The detection of OpposingFlow and PersonRuns events was tested and analyzed as an approach to detect events based on motion analysis. Use of grayscale images was useful for simplicity in the implementation and definition of zones where events should occur probed to be effective to discard undesired detections. After the analysis of the background subtraction and optical flow techniques, approximate median and Lucas-Kanade were chosen respectively. The segmentation of blobs gave shapes far from ideal representation of semantic objects, but they were useful to determine the magnitude and direction using the correspondent optical flow vectors belonging to the blob's area. Magnitude of optical flow vectors was the key to determine blobs running, and the angle of optical flow vectors was the key to determine blobs moving in specific directions. Detected blobs matching the criteria were counted for every frame and then filtered to

determine whether the event was or not present. Results from a test performed on a set of videos give a precision of 64.28% at 81.8% of recall for OpposingFlow events and a precision of 50% at 81.8% of recall for PersonRuns events. TRECVID 2008 participants reported precision values between 1.85% - 7.5% at recall values between 75% - 81% for OpposingFlow events[17,18,19,24], and precision values between 1.9% - 5.9% at recall values between 26% - 45% for PersonRuns events[18,19,22,24]. The big difference between some of the values compared to TRECVID is because TRECVID reports include all the results from the five different cameras in the dataset and this work used only videos from one camera, moreover, the small video clips used have reduced number of false detections compared with a system analyzing larger videos.

## **Chapter 6 CONCLUSIONS AND FUTURE WORK**

This chapter presents conclusions and directions for future work.

### **6.1 Conclusions**

This thesis presented a working system for event detection in surveillance video focusing on the detection of 2 specific events: OpposingFlow and PersonRuns.

The proposed system is capable of:

- Determining the magnitude and direction of movement of the objects detected in the foreground.
- Detecting OpposingFlow events.
- Detecting PersonRuns events.

On the other hand, the system is subject to the following restrictions:

- Use of a fixed camera setting.
- The objects in the foreground need a minimum texture for a correct optical flow calculation.

- The system needs a minimum contrast between the foreground objects and the background in order to work. Acceptable values of difference between foreground and background must be greater than 20.
- The current implementation is used only to detect 2 specific events: OpposingFlow and PersonRuns. These events depend on motion across the scene. Detection for other different events requires redesign in different blocks of the system.
- Duration of events must be greater than a second to be detected.
- Children and small people cannot be detected for OpposingFlow event because they are usually outside the ROI.
- Memory limits affect the amount of information that can be processed with the current implementation.

Through study and experimentation, this work has reached the following conclusions:

- The introduction of the event probability zone detection improves the response of the system due to the suppression of undesired object's behaviors as well as the reduction in time and computational resources because of the limited amount of information in the ROI.
- The approximate median method for background subtraction turned out to be the most suitable option because it is fast, easy to implement and handles noise relatively well. Although frame difference method is also fast and easy to implement, it is very susceptible to noise and highly dependent on continuous

movement of the objects. Finally, the Mixture of Gaussians handles noise relatively well but its processing time is large and the numerous parameters for configuration make the method very difficult to tune.

- The Lucas-Kanade method for the optical flow evaluation turned out to be the most suitable option because it outperforms the processing time compared to the Horn-Schunck method. Moreover, the Lucas-Kanade method has only one parameter to configure with the possibility to change the density of vectors while the Horn-Schunck method has two parameters to configure and the optical flow is evaluated for every single pixel in the image leading to higher density of vectors.
- Morphological operations proved to be a valuable method to remove noise after the background subtraction process. They were also useful for the segmentation of blobs to mask the optical flow vectors with higher contrast in the image. Even though, resulting blobs are far from ideal because they are not a good representation of semantic objects.
- The solution using small video clips was an approach that affected the solution's performance, specifically when calculating the precision for detections. Presence of more false detections is expected when using longer videos.
- The minimum distance used to filter the data in the post-processing stage having two purposes: distance between different events as well as the duration of every event, should be separated as two different parameters for a better implementation of the system.
- Information such as relative velocity, area and position of detected objects can be extracted from the system.

- Events such as OpposingFlow and PersonRuns can be detected with a moderate precision with the system implementation.
- Modularity proved to be an important approach to the development of the system. It helped to implement and evaluate different algorithms independently as well as two different event detections. MATLAB was very useful to develop the system but for better performance in the implementation, the use of lower level languages is required as well as optimization of the code.

## **6.2 Future Work**

- Additional computer vision techniques can be combined to the proposed solution in order to increase the confidence level in detections. An implementation using color based techniques among others can help to make a more robust system.
- A more complex occlusion handling approach can be implemented using tracking techniques, probability and predictors.
- Optimization of the code, migration to other programming language, or hardware implementation must be taken into account for a better performance of the system.
- Implementation of the system using MPEG-7 descriptors to represent the objects and its features. An event detection approach can use the information of the MPEG-7 descriptors to help with the automatic identification of events. Using the features extracted by MPEG-7, information about elements and the interaction among them can be used to analyze the presence of an event and increase the



confidence in different event detections. Supposing the video surveillance system has cameras with MPEG-7 preprocessing capabilities to extract objects and its features such as color, shape, contours, motion, etc, the information would be stored as metadata attached to the media and could be used to be reviewed offline by an operator for further analysis of detections. In this case, it is important to consider the byte overhead in the multimedia file due to the metadata attached to the video. As a preliminary evaluation,

- Table 10 shows the most common MPEG-7 descriptors suitable for event detection and their size to estimate the byte overhead for every frame[13,14].

**Table 10: Byte overhead for MPEG-7 Descriptors**

<b>GROUP</b>	<b>DESCRIPTOR</b>	<b>SIZE PER FRAME (Range in Bytes)</b>
COLOR FEATURE DESCRIPTORS	ColorLayout	8
	ColorStructure	32 - 256
	DominantColor	6 - 34
	ScalableColor	4 - 128
	GoFGoPColor	4 - 128
TEXTURE FEATURE DESCRIPTORS	EdgeHistogram	30
	HomogeneousTexture	33 - 63
	TextureBrowsing	1 - 2
SHAPE FEATURE DESCRIPTORS	RegionShape	18
	ContourShape	4 - 76
MOTION FEATURE DESCRIPTORS	MotionActivity	1 - 104
	MotionTrajectory	Additional Dependencies
	ParametricObjectMotion	Additional Dependencies

## BIBLIOGRAPHY

- [1] T. P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, and V. Eruhimov, "Computer Vision Workload Analysis: Case Study of Video Surveillance Systems", Intel Technology Journal, 9(12), May 2005.
- [2] W. Hu, T. Tan, L. Wang, and S. Maybank, "Survey on Visual Surveillance of Object Motion and Behaviors", *IEEE Transactions on systems and cybernetics*, Vol. 34, No. 3, pp. 334-353. August 2004.
- [3] Haritaoglu, D. Harwood, and L. S. Davis, "W<sup>4</sup>: Real-time surveillance of people and their activities," IEEE Trans. Pattern Anal. Machine Intell., vol. 22, pp. 809–830, Aug. 2000, pp.809-830.
- [4] M. Piccardi, "Background subtraction techniques: a review," Systems, Man and Cybernetics, 2004 IEEE International Conference on , vol.4, Oct. 2004, pp. 3099-3104.

- [5] H. Z. Sun, T. Feng, and T. N. Tan, "Robust extraction of moving objects from image sequences," in Proc. Asian Conf. Computer Vision, Taiwan, R.O.C., 2000, pp. 961–964.
- [6] B. Horn and B. Schunck, "Determining Optical Flow", *Artificial Intelligence*, Vol. 17, pp. 185-203, 1981.
- [7] B. Lucas, and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proc. DARPA IU Workshop*, pp. 121-130. 1981.
- [8] J. Barron, D. Fleet, and S. Beauchemin, "Performance of Optical Flow Techniques", *Int. J. Comput. Vis*, Vol. 12, pp. 42–77. 1994.
- [9] N. Friedman and S. Russell, "Image segmentation in video sequences: a probabilistic approach," in Proc. 13th Conf. Uncertainty in Artificial Intelligence, 1997, pp. 1–3.
- [10] A. Murat Tekalp, "*Digital Video Processing*", Prentice Hall PTR, Upper Saddle River, NJ: 1995.
- [11] R. C. Gonzalez and R. E. Woods: "Digital Image Processing", Prentice Hall, Second edition (2002).

- [12] B. S. Manjunath, P. Salembier, & T. Sikora, (Eds). (2002, April). Introduction to MPEG-7: Multimedia content description interface. New York: Wiley Publications
- [13] <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm#E9E1>
- [14] ISO/IEC 15938-3: 2001, “Multimedia content description interface – Part 3 Visual”, Version 1.
- [15] A. Stergiou, A. Pnevmatikakis, L. Polymenakos, “Detecting Single-Actor Events in Video Streams for TRECVID 2008”, Athens Information Technology and Aalborg University, CTiF, TRECVID 2008.
- [16] Mert Dikmen, Huazhong Ning, Dennis J., “Surveillance Event Detection”, TRECVID 2008.
- [17] P. Chmelař, V. Beran, A. Herout., “Brno University of Technology at TRECVID 2008”. TRECVID 2008.
- [18] Sheng Tang, Jin-Tao Li, Ming Li, Cheg Xie., “TRECVID 2008 Participation by MCG-ICT-CAS”, Chinese Academy of Sciences. TRECVID 2008
- [19] P. Wilkins, P. Kelly, C. Ó Conaire., “Dublin City University at TRECVID 2008”, Dublin City University. TRECVID 2008.

- [20] X. Xue, W. Zhang, Y. Guo, H. Lu., “Fudan University at TRECVID 2008”. Fudan University. TRECVID 2008.
- [21] P. Yarlagadda, K. Garg, S. Guler., “Intuvision Event Detection System for TRECVID 2008”. IntuVision, Inc. TRECVID 2008.
- [22] M. Taj, F. Daniyal, A. Cavallaro., “Event analysis on TRECVID 2008 LondonGatwick dataset”. Queen Mary, University of London. TRECVID 2008
- [23] K. Yokoi, H. Nakai, T. Sato., “Toshiba at TRECVID 2008: Surveillance Event Detection Task”. Corporate Research and Development Center, TOSHIBA Corporation. TRECVID 2008.
- [24] O. Orhan, J. Hochreiter, J. Pooch, Q. Chen, A. Chabra, M. Shah., “University of Central Florida at TRECVID 2008 Content Based Copy Detection and Surveillance Event Detection”. University of Central Florida. TRECVID 2008.
- [25] “TRECVID 2009 Event Annotation Guidelines”. Version 1.0. June 2, 2009
- [26] Gonzales, Woods, Eddins, (2002) “Digital Image Processing using MATLAB“, Prentice Hall, ISBN 0130085197.

- [27] MATLAB Help.
  
- [28] C. Pertuz, “Methods and Algorithms for Human Detection in Video Sequences “,  
Master thesis defense, Florida Atlantic University, 2007.
  
- [29] D. Pava, “Object detection in Low Resolution Video Sequences”, Master thesis  
defense, Florida Atlantic University, 2009.
  
- [30] A. Fonseca, “A Study on Implementing Autonomous Video Surveillance Systems  
Based on Optical Flow Concept”, Master thesis defense, Florida Atlantic  
University, 2007