COMPRESSION-BASED ANOMALY DETECTION FOR VIDEO SURVEILLANCE APPLICATIONS

Carmen E. Au

Department of Electrical and Computer Engineering

McGill University, Montréal

February 2006

A thesis submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of
Master of Engineering

Your file Votre référence
ISBN: 978-0-494-24936-9
Our file Notre référence
ISBN: 978-0-494-24936-9

Abstract

In light of increased demands for security, we propose a unique approach to automated video surveillance using anomaly detection. The success of this approach is dependent on the ability of the system to ascertain the novelty of a given image acquired by a video camera. We adopt a compression-based similarity measure to determine similarity between images in a video sequence. Images that are sufficiently similar to the previously-seen images are discarded; conversely, images that are sufficiently dissimilar are stored for comparison with future incoming images.

The use of a compression-based technique inherently reduces the heavy computational and storage demands that other video surveillance applications typically have placed on the system. In order to further reduce the computational and storage load, the anomaly detection algorithm is applied to edges and people, which are image features that have been extracted from the images acquired by the camera.

Résumé

Afin de satisfaire la demande croissante en matière de sécurité, nous proposons une approche unique de surveillance visuelle automatisée basée sur la détection d'anomalies. Le succès de cette approche repose sur la capacité du système de détecter une nouveauté dans une image donnée, acquise par caméra vidéo. Une méthode de mesure de similitude, basée sur la compression, est utilisée pour déterminer la similarité entre des images acquises lors d'une séquence vidéo. Les images qui sont suffisamment semblables aux images précédemment observées sont rejetées. Contrairement, les images qui sont suffisamment différentes sont sauvegardées pour être comparées aux images qui seront prochainement acquises.

L'utilisation d'une technique basée sur la compression réduit les charges de calcul et de mémoire, généralement requise par d'autres applications de surveillance visuelle. Dans le but d'optimiser la réduction de ces charges, l'algorithme de détection d'anomalies est appliqué sur deux charactéristiques des images acquises: les bords et les personnes.

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to Prof. James J. Clark. He has guided and supported me throughout my graduate studies. Above all, he has always treated me with kindness and respect. My time spent under his mentoring has truly been enriching.

Danny Chouinard, Jan Binder, Marlene Grey, and Cynthia Davidson have been incredibly supportive. They, together with all the members of CIM, professors, students and staff alike, have made my graduate studies wonderfully enjoyable.

The Visual Motors Research Lab has been a great working environment. Sandra Skaff has always been generous with her time and support, and Trevor Ahmedali was the best officemate I could ask for. I have learned a lot from both of them.

Many thanks belong to my army of support, Franco P., Sadok A., Usman K., Zahra A. and the rest of the "boys upstairs", for bolstering my confidence and keeping me laughing. And special heartfelt thanks to PKC and the Ho Family for feeding me, and my hilarious godbrothers for reminding me how to laugh at myself.

I would also like to thank Elijah for sticking by me. His constant belief in me sustained me during these past 2 years. I will forever be grateful to him.

Finally, I would like to thank my family, who mean everything to me. My parents' and sister's continual love and pride in me are more precious to me than any accolade. I love them all dearly. I could not have imagined or prayed for a better family.

Table of Contents

Abstract i

Résumé ii

Acknowledgements iii

List of Figures vi

Glossary ix

CHAPTER 1 Introduction 1

1.1 Automated Video Surveillance Using Anomaly Detection 2
1.2 Our Demonstration system 3
1.3 Our Approach 5
1.4 Thesis Overview 7

CHAPTER 2 Background 9

2.1 Video Surveillance Methods 9
2.1.1 Object (person) Detection: *9*
2.1.2 Object Tracking *10*
2.1.3 Movement/Activity Classification *11*
2.2 Edge Detection Methods 11
2.3 Compression Methods 14
2.3.1 Entropy coding *15*
2.3.2 Ziv-Lempel algorithms *19*
2.3.3 JPEG compression *19*
2.3.4 Other universal compression algorithms *20*

CHAPTER 3 Anomaly Detection on Intensity Images 22

3.1 *bzip2* compression scheme 23
3.1.1 The Burrows-Wheeler Transform *23*
3.1.2 Move-To-Front coding *25*
3.1.3 Huffman coding *25*
3.2 Our Anomaly Detection Algorithm 28
3.3 Results 30

CHAPTER 4 Anomaly Detection on Edge Images 40

4.1 Integrating the Canny Edge Detector 40
4.2 Results 42

CHAPTER 5 Anomaly Detection with Person Detection 50

5.1 The Nair and Clark Person Detector 50
5.2 Results 53

CHAPTER 6 Conclusion 62

6.1 Overall Limitations and Possible Improvements 62
6.2 Future Applications and Conclusion 64

References 66

List of Figures

Figure 1.1. Axis 200+ netcam 3

Figure 1.2. Corridor being monitored with our video surveillance system. 4

Figure 1.3. Block diagram for anomaly detection using similarity measure. 6

Figure 1.4 Example images acquired from our demonstration system. 7

Figure 1.5. Edge images of the example images from Figure 1.4. 7

Figure 1.6. Person detection applied to example images from Figure 1.4. 7

Figure 2.1. Step edge with first and second order derivative. 12

Figure 2.2. Canny edge detector algorithm. 14

Figure 2.3. Block diagram for arithmetic coding. 16

Figure 2.4. Table of probabilities and ranges. 16

Figure 2.5. Arithmetic coding algorithm 17

Figure 2.6. Arithmetic coding for "abracadabra". 17

Figure 2.7. Huffman coding for "abracadabra" 18

Figure 2.8. Block diagram for JPEG compression techniques. 20

Figure 3.1: BWT applied to "wheelerwheeler". 24

Figure 3.2. MTF for "wheelerwheeler" with 14 byte block-size. 25

Figure 3.3. Huffman coding for "wheelerwheeler" with 14 byte block-size. 26

Figure 3.4. BWT for "wheelerwheeler" with 9-byte block-size 27

Figure 3.5 Pseudocode for anomaly detection algorithm. 29

Figure 3.6. Highest similarity measure for each intensity image processed over 2 days 31

Figure 3.7. Highest similarity measure for each intensity image processed over 1 day 32

Figure 3.8. Number of novel images vs. number of intensity images processed over 6 weeks...34

Figure 3.9. Number of novel images vs. number of intensity images processed over 1 day.....35

Figure 3.10. Examples of similar (top) and dissimilar (bottom) image classification. 36

Figure 3.11. Example of a false positive: two similar images classified as dissimilar. 37

Figure 3.12. Processing time for each novel intensity image. 39

Figure 3.13. Processing time for each novel image vs. total number of images processed.....39

Figure 4.1. Pseudocode for anomaly detection on edge images 41

Figure 4.2. Example of an intensity image and its corresponding edge image.....41

Figure 4.3. Highest similarity measure for each edge image.....43

Figure 4.4. Number of novel images vs. number of edge images processed over 5 days. 44

Figure 4.5. Number of Novel Intensity and Edge images vs Number of Images Processed.....45

Figure 4.6. Highest similarity measure for intensity and edge images 45

Figure 4.7. Example images from Figure 3.11 classified correctly due to edge detection. 46

Figure 4.8. Processing time for each novel edge image. 49

Figure 4.9. Processing time for each novel image vs. total number of images processed.....49

Figure 5.1. Background subtraction of example images.. 51

Figure 5.2. Scan window moving across image, classifier classifying each subimage as "person" or "non-person" 51

Figure 5.3. The basic structure of the Nair and Clark person detector. [1] 52

Figure 5.4. Block diagram of anomaly detector with person detection.....52

Figure 5.5. Highest similarity measure for each person image.....53

Figure 5.6. Number of Person Images vs Number of Images Processed 54

Figure 5.7. Number of Novel Images vs Total Number of Images.....54

Figure 5.8. Figure from location invariance test. 57

Figure 5.9. Cropped people from location invariance test. 57

Figure 5.10. Examples of false positives for the Nair and Clark person detector.....58

Figure 5.11 Intensity, edge and person image sizes. 59

Figure 5.12. Processing time for each novel person image 60

Figure 5.13. Time to process intensity, edge and person images 61

Glossary

BWT	Burrows-Wheeler Transform
CGI	Common Gateway Interface
CPU	Central Processing Unit
DMC	Dynamic Markov coding
DVR	Digital Video Recorder
FSM	Finite-state machine
GDMC	Generalized dynamic Markov coding
GIF	Graphics Interface Formats
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Expert Group
LZ	Ziv-Lempel Compression Algorithms
PDAD	Person detector with anomaly detection algorithm
PPM	Partial matching algorithm
TIFF	Tagged Image File Format
URL	Uniform Resource Locator
VSAM	Video Surveillance and Monitoring

CHAPTER 1

INTRODUCTION

The heightened need for security has increased the desire for fast, reliable, and low-cost video surveillance. The objective of video surveillance is to monitor an indoor and/or outdoor environment; as well, to report any information about relevant and often suspicious activity. Present surveillance systems consist of aptly placed cameras that capture the activity of a given scene, and television screens that display the video streams acquired by the camera. Security guards watch the television screens to determine if there is any activity that warrants a response. Given that these events rarely occur, more often than not, it is the security guard who stares into a field of monotonous scenes; vigilance is thus very difficult to maintain. In a security system, where vigilance is of utmost importance, the burden of detecting these anomalous events cannot be left upon the security guard. Therefore, the goal of this work is to remove the onus of detecting these anomalous events from the guard, and to place it on the surveillance system.

In order to detect one of these events, the surveillance system must have the ability to differentiate between regular, routine, and novel events. Like the security guard, giving the surveillance system that ability is not a trivial task. The question we are essentially posing the system is, "has this event been seen before"?

Anomaly detection is the process where a baseline model of behaviour or activity is established, and any deviation from this baseline triggers an alert. In computer security applications, the computer system itself is monitored for deviating behaviour [19, 45, 52, 71, 75, 80, 88]. In health-related applications, anomaly detection is used to detect disease outbreaks [20, 79, 83-85]. In video surveillance, the activity within a given environment is monitored for deviations.

1.1 Automated Video Surveillance Using Anomaly Detection

Our primary objective is to build an automated video surveillance system capable of anomaly detection. The anomaly detection should be performed by the system itself without any user input. In addition to this objective, there are certain criteria desirable in a surveillance system which we used as a guideline:

1) **Real-time processing:** The system should be able to detect for anomalies in real-time, or as close to real-time as possible.

2) **Minimize the computational load on the system:** The effectiveness of the surveillance system is highly incumbent on the speed of the algorithm, thus, the computational load on the system should be minimized.

3) **Adaptability:** The baseline of normal behaviour in the environment being monitored should be adaptable to the changes often seen in the real-world. For example, clothing changes from season to season and even from day to day. Moreover, someone new should be able to be added to the system without need for retraining. This last point is in congruence with the next criterion.

4) **Online training:** The system should have very little to no manual training. Manually training the system is tedious and time-consuming, and as a real-world application, this would be inefficient. Moreover, as aforementioned, should there be new people or new activity that need to be introduced to the baseline, online training would enable the system to integrate the new information with the database without stopping the system.

5) **Minimize the storage load on the system:** Because the system requires a store of past knowledge to determine the level of novelty of each image, a large database will be required. In order to meet with the other requirement of adaptability, this database must continue to grow, and in real-world

applications, storage space will become a concern. Thus, one criterion will be to minimize the amount of storage needed.

6) **Scalability:** Since many surveillance systems use a network of cameras to monitor multiple sites or a single site from different views, the system should be able to introduce new cameras into the network with ease.

1.2 Our Demonstration system

For our demonstration system, we use a netcam, the Axis 200+, shown in Figure 1.2, to monitor the corridor in McGill University's Centre for Intelligent Machines. A netcam is a video camera where the latest image acquired can be downloaded from the World Wide Web. Our netcam has a built-in Hypertext Transfer Protocol (HTTP) server where the images downloaded from the Internet are in Joint Photographic Expert Group (JPEG) format.

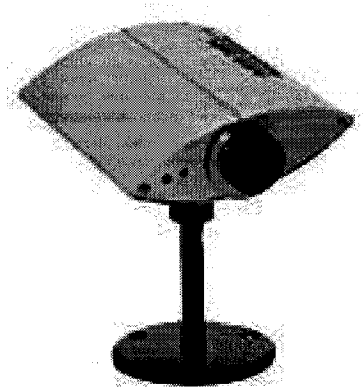

Figure 1.1. Axis 200+ netcam

The camera server takes a snapshot every time an `http` call is made. The uniform resource locator (URL) is in the following format:

```
http://<IP Address>/<snapshot type>.jpg?[<parameter>=<value>…]
```

Figure 1.2. Corridor being monitored with our video surveillance system.

The snapshot type and the common gateway interface (CGI) parameters can be specified. From the three snapshot types listed in Table 1.1, we see that there is an obvious tradeoff between size and time. While the largest image size would capture more details for the anomaly detection, at a rate of 18 seconds/frame, the acquisition time is far too long for a real-world surveillance system. The halfsize option, on the other hand, has the best rate of acquisition; however, it would be at the expense of image quality. Thus, we chose the fullsize image, which acquires images at 0.5 seconds/frame. Unfortunately, this rate implies that the surveillance system is limited by that rate of acquisition, thus regardless of the speed of the algorithm, the maximum processing rate is roughly 2 frames/second.

Snapshot Type	Size (pixels)	Acquisition Time
fullsize.jpg	352x288	0.5 seconds/snapshot
halfsize.jpg	176x144	0.3 seconds/snapshot
hugesize.jpg	704x576	18 seconds/snapshot

Table 1.1 Three main snapshot types available for the Axis 200+ netcam.

The CGI parameters available for adjusting are listed in Table 1.2 with the values to which we set our demonstration camera in bold. The compression parameter specifies the quality of the image. Again, we have a tradeoff between quality and speed. Thus, we chose a "medium" compression, which is the default camera setting.

Parameter	Values
compression	[medium | high | low]
color	[normal | none]
mirror	[on | off]
clock	[show | hide]
rotation	[normal | upsidedown | 90deg | 270deg]

Table 1.2 Some CGI parameters and values for the Axis 200+ netcam.

One major benefit of using a netcam is that adding cameras to the surveillance system is done by simply connecting the additional cameras to the Internet. The system can thus be distributed over several cameras that communicate over the Internet. Moreover, the physical distance between the cameras and the computer, which processes the images, can be as large as the farthest distance between any two points on the network [58]. Finally, netcams are low-cost and easy to use.

1.3 Our Approach

Our video surveillance system applies a compression-based anomaly detection algorithm based on the technique described by Bennett *et al.* [5] to classify chain letters. They devised a similarity measure stemming from the notion that the compression of a data file provides a good measure of its information content. A compression algorithm

seeks to remove all redundancies within a data file in order to save storage space on the hard disk. When two files are compressed together, if the two files have no redundancies between them, then the compression of the joint files will be as big as the sum of the compression of the two individual files. However, if the two files contain redundancies, then the compression of the joint files will be smaller than the sum of the compression of the two individual files. Finally, with an ideal compressor, if the two files are identical, then the compression of the joint files will be as big as the size of the compression of one instance of the file. Thus, the size of the joint compressed files compared to the size of the individual compressed files combined, is a good measure of the files' similarity.

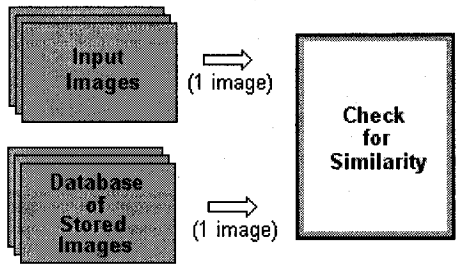

Figure 1.3. Block diagram for anomaly detection using similarity measure.

Anomaly detection is achieved by comparing all images acquired from the surveillance system, and comparing them to a database of stored images. In Figure 1.4, examples of intensity images of the scene under surveillance are shown. We first apply the anomaly detection algorithm on the intensity images. Subsequently, we apply the algorithm to various features of the images, namely: edges and persons. Edge detection is the process, whereby, only the boundaries of the discontinuities that exist in image intensity are preserved, and the rest of the image is discarded. Figure 1.5 shows the example images from Figure 1.4 after an edge detector has been applied.

Figure 1.4 Example images acquired from our demonstration system.

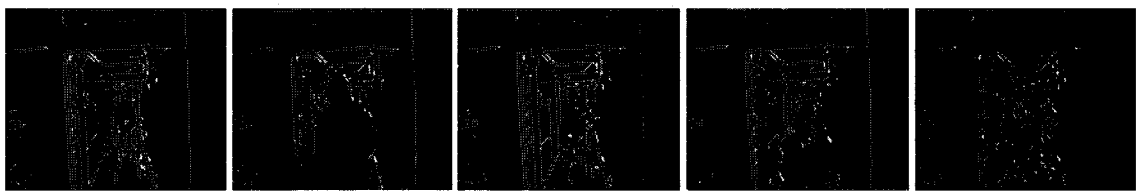

Figure 1.5. Edge images of the example images from Figure 1.4.

Since, for the most part, video surveillance involves monitoring the activity of people, a person detector is first applied to the sequence of images. In computer vision, a person detector detects the number and location of persons, if any, in a given scene. Their location is generally specified by drawing the smallest rectangular bounding box that encompasses the person(s). The people detected from the intensity images in Figure 1.4 are shown in Figure 1.6. It is these bounding boxes that are input into the anomaly detector.

Figure 1.6. Person detection applied to example images from Figure 1.4.

1.4 Thesis Overview

This thesis describes the anomaly detection algorithm we have developed for the purpose of video surveillance. Chapter 2 begins by summarizing the main classes of video surveillance techniques, including the person detector we implemented. The

different edge detection methods are then presented, followed by an explanation as to why we chose the particular edge detector that we implemented. Finally, a brief overview of different compression techniques is presented.

In chapter 3, the compression technique we employed, as well as the anomaly detector that we implemented are explained in greater detail. We then present the results of some tests that we ran on intensity images, which were acquired from our demonstration system described in section 1.2. The focus of these tests was to determine how closely our algorithm adhered to the criteria of a surveillance system presented in chapter 1.

While our approach is successful in detecting anomalous intensity images, there were certain shortcomings; namely, the algorithm was sensitive to illumination and location changes. Thus, in chapter 4 and 5, we present the work that we did on applying edge detection and person detection to the images respectively, in order to extract features that would help overcome the limitations of the anomaly detector.

CHAPTER 2

BACKGROUND

2.1 Video Surveillance Methods

Extensive research has been dedicated to developing video surveillance techniques. Of the plethora of existing techniques, the main techniques can be divided into three main tasks: object detection, object/region tracking, and activity/movement classification, defined in Table 2.1. Many of the video surveillance methods perform at least one of these tasks.

Task	Definition
Object Detection	The location and often orientation of objects of interest in a given scene are specified.
Object Tracking	Objects of interest in a given scene are detected and pursued during subsequent frames using motion information.
Movement/Activity Classification	Motion or the statistical sequences of movements of objects of interest are classified.

Table 2.1. Three main video surveillance tasks.

2.1.1 Object (person) Detection:

Object detection is the task of finding a specified object, if it exists, in a given scene. For most surveillance systems, people are the target category of interest. In any given scene, there could be zero, one or many persons. Person detection can be divided into three main classes: model-based detection, motion-based detection and appearance-based detection. Model-based methods [25, 26, 44] find a person in an image using a parts-based person model to detect body parts which are then assembled into a full

person. The benefit of this technique lies in the non-rigidity of the model; the persons detected are not limited to one pose.

Because people are constantly moving, motion-based methods [22, 33, 72, 86] use this movement as their primary source of information to detect for people. One popular approach is the use of background subtraction, where a background model is computed, and all subsequent images are compared against the background model. The pixels which differ from the background model, by a predefined threshold, are considered foreground, or people.

Appearance-based methods [27, 60, 61, 77] uses a two-class classifier to distinguish cropped images of people from all other types of images. These methods do not require *a priori* information about the human body's structure, but rather learn from examples.

In our approach, we apply a person detector to the images before applying the anomaly detector. A method that was consistent with the criteria which we set in chapter 1 was needed. One such method was Nair and Clark's appearance-based person detector [57, 58]. Their person detector employs a classifier that is learned online. Training examples acquired directly from the camera are automatically labeled, and these examples are used to train the classifier. Thus, the learning can be performed in parallel with the detection, and the classifier is adapted over time. The Nair and Clark person detector overcomes the limitations of manual labeling and offline training present in other object/person detection with learned classifiers [60, 76, 77]. For this reason, we chose to adopt their person detector for our anomaly detection scheme.

2.1.2 Object Tracking

Tracking techniques can be divided into two categories of tracking: recognition-based tracking [28, 50, 87] and motion-based tracking [3, 7, 41, 55]. In recognition-based tracking, regions in a given scene that match a pre-defined object model are identified as objects of interest; these objects are then pursued. The main advantage of

this approach is that object tracking can be achieved in three dimensions, and the objects of interest's rotation and translation can be estimated. The main disadvantage is that *a priori* knowledge of the object model is required before tracking can be performed. Motion-based tracking eliminates the need for the object model, by using motion parameters to detect for objects. In this case, regardless of shape or size, the objects can be tracked.

There is a fair amount of overlap between the work done on object detection and that done on object tracking. For example, in [33], Haritaoglu *et a.l* proposed the W^4 system which detects, *then* tracks human heads, torsos, arms and legs of upright people in real-time. In fact, the task of object tracking implies that object detection must first occur.

2.1.3 Movement/Activity Classification

Movement or activity classification is the process of classifying and recognizing observed motion or the statistical pattern of the observed motion. The VIEWS system, developed by Corrall, attempts to describe activity in a scene using a model-based approach [17]. The system required a large amount of *a priori* knowledge of the camera models, ground plane representation, 3-D object models, and behavior models [21]. VIEWS is one of the earliest attempts at activity classification. From this system, the PASSWORD system was spawned, which is essentially a low cost parallel digital signal processor version of VIEWS [6, 14]. Later work on movement or activity classification, include the Video Surveillance and Monitoring (VSAM) systems created by Carnegie Mellon University [42], MIT [40] and Texas Instruments [24]. Grimson *et al.* also developed a system that achieves object and activity classification at a low level [31].

2.2 Edge Detection Methods

Edges in images represent the discontinuities in the image intensity due to the changes in structure. These discontinuities in image properties usually reflect important

events and changes in world properties. In computer vision, the purpose of edge detection is to filter out a significant portion of the image, while preserving the structural properties within it. These properties include the photometrical, geometrical, and physical characteristics of objects which vary, in order to give the different grey levels in images.

The main types of edges are step edges, line edges [2, 13, 30, 67, 70], and junctions [46, 54, 59, 65]. Line edges are the local extrema of the grey level image. They are useful in detecting roads and rivers in images. Junctions are regions in the image where two edges meet, and are helpful in solving the correspondence problems in computer vision. The edges that we are concerned with in this thesis are step edges, which are step discontinuities within the image. In Figure 2.1 an ideal step edge with its first and second derivative is shown.

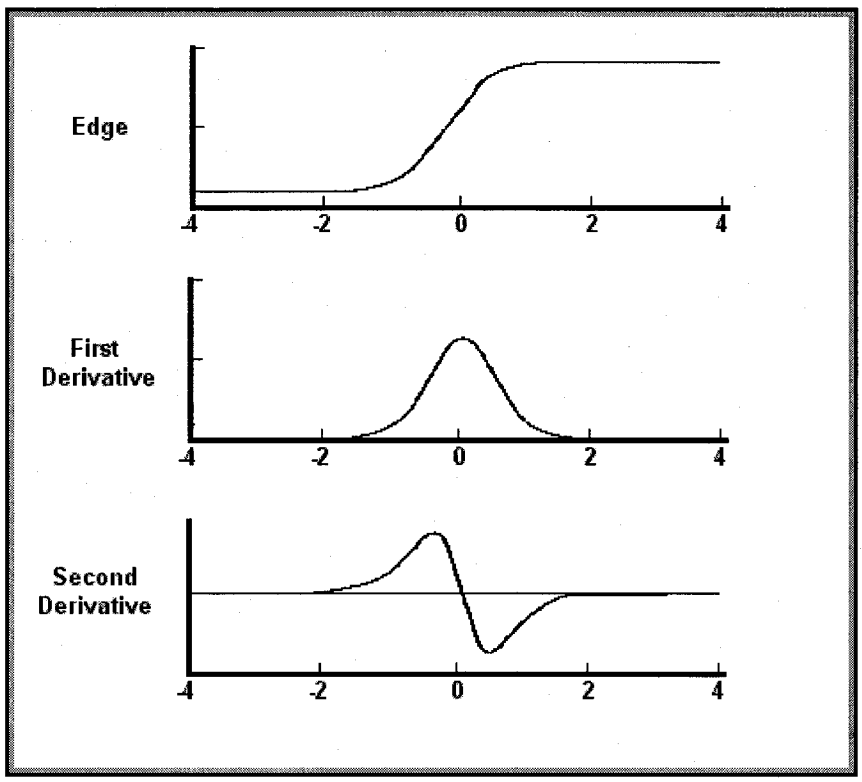

Figure 2.1. Step edge with first and second order derivative.

Based on the notion that edges occur at discontinuities in luminance in the image, by taking the derivative of the intensity values across the image, the edges will be at the point where the derivatives are maximum or minimum. Two types of edge detectors were developed from this idea: the gradient method, which takes the first derivative of the image, and the Laplacian method which takes the second derivative.

The gradient method detects edges by finding the maximum or minimum in the derivative of the intensity values across the image. In this method, the intensity image is convolved with a 3x3 mask, in order to approximate the first-order partial derivatives. In order to detect both the horizontal and vertical edge component, the images are, in fact, convolved with a mask along the rows and along the columns. The masks for the Sobel [38, 43] and Prewitt [62] step edge detectors are:

$$\Delta_x = \begin{bmatrix} -1 & 0 & 1 \\ -a & 0 & a \\ -1 & 0 & 1 \end{bmatrix} \qquad \Delta_y = \begin{bmatrix} -1 & -a & -1 \\ 0 & 0 & 0 \\ 1 & a & 1 \end{bmatrix}$$

where a is 2 for Sobel and 1 for Prewitt edge detectors.

The Laplacian method finds zero crossings in the second derivative of the image in order to locate the edges. A common mask for Laplacian edge detectors is:

$$\nabla = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The performance of these detectors deteriorates as the noise in the images increase. Rosenfeld and Thurston [66] introduced the concept of smoothing to reduce the noise in the image. In smoothing, a pixel intensity value is replaced by the average of the neighbouring pixels. While the smoothing process reduces the amount of noise in the image, it also causes a loss of information. Thus, many attempts have been made to design an edge detector which balances this tradeoff [53, 68].

The Canny edge detector [12] is a popular edge detector with the three-fold goal of:

1. Maximizing the signal to noise ratio for better edge detection
2. Achieving good localization to accurately mark edges, and
3. Minimizing the number of responses to a single edge. In other words, non-edges are not marked.

Because of it these characteristics, the Canny edge detector was thus selected, in this thesis, as the edge detector applied to the images before anomaly detection. In Figure 2.2, a brief outline of the Canny edge detector is listed. The way in which the edge detector is integrated with our algorithm, is explained in chapter 3.

```
1: Smooth the image with a Gaussian filter
2: Compute the gradient magnitude and orientation
3: Apply nonmaxima suppression
4: Use double thresholding to detect and link edges
```

Figure 2.2. Canny edge detector algorithm.

2.3 Compression Methods

Data compression reduces the size of a file, which can lead to a reduction in computing resources, transmission time, or storage resources. Compression techniques can be reduced to two main types: lossy and lossless. In lossy type compression schemes, the original file cannot be reproduced from the compressed file. On the other hand, with lossless methods, no information is lost during the compression process and the original can be reproduced.

Modern compression can be split into two stages: modeling and coding. In the modeling stage, the similarities and regularities within the sequence to be compressed are found. In the coding stage, these redundancies are eliminated. While some older compression methods, such as the Ziv-Lempel algorithms [89, 90], are not split into the

two stages, for the most part, the two stage approach is the modern paradigm of compression. In the next section, we will present the two main coding methods which many of the compression algorithms use. In Section 2.3.2, we will discuss the classic Ziv-Lempel algorithm, and subsequently, will present the more recent approaches to compression.

2.3.1 Entropy coding

The second stage of compression, the coding stage, is where compression actually takes place. In entropy coding, sequences are compressed based on a probability distribution of occurrences of the alphabet symbols, which is determined in the modeling stage. Based on the probability of a given character appearing in the sequence, the character is assigned a code - whose length is proportional to the negative logarithm of that probability. In other words, each character is assigned a code of length:

$$\log\left(\frac{1}{p_i}\right) = -\log(p_i)$$

where p_i is the probability of occurrence of the character. Thus, the characters with the highest probability of being in the sequence, the most common characters, use the shortest codes. Two main entropy encoding algorithms are the arithmetic and Huffman coding.

2.3.1.1 Arithmetic coding

Arithmetic coding [36, 47, 63, 64] is a form of entropy encoding which codes the entire input message into a single number that ranges from 0.1 to 1.0. The algorithm begins by computing the frequency of each character in an input sequence. Based on the characters' frequency, a probability table is created and a range between 0 and 1 is assigned to each input character. The range consists of a low value and a high value. The order of assigning these ranges is not important; however, both the encoder and decoder must follow the same order. And thus, the more frequent the characters, the

wider the range assigned to it. Once each instance of each character in the sequence is assigned a range, the encoding can begin.

The encoding begins with a range between 0.0 and 1.0. The first character in the input sequence constrains the output number with its corresponding range. The range of the next character will then further constrain the output. The output continues to be constrained as each character in the input sequence is processed. The greater the number of input characters in the sequence, the more precise the output number will be.

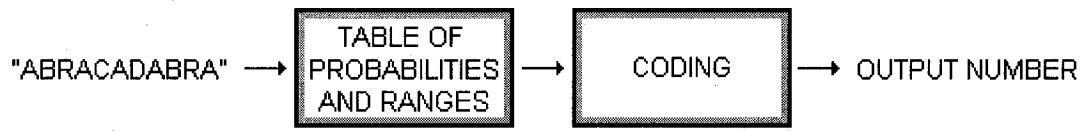

Figure 2.3. Block diagram for arithmetic coding.

In Figure 2.3, the string "`abracadabra`" is input to the coding algorithm. The arithmetic encoder has two main steps: building the table of probabilities and ranges, and coding. In Figure 2.4, the probability of each character in "`abracadabra`" is determined and assigned the appropriate range. The range of each character has a low and high value, which we will refer to as `CHAR_LOW` and `CHAR_HIGH` respectively.

	PROBABILITY	CHARACTER RANGE
A	$\frac{5}{11} = 0.45455$	[0, 0.45455)
B	$\frac{2}{11} = 0.18182$	[0.45455, 0.63636)
C	$\frac{1}{11} = 0.09091$	[0.63636, 0.72727)
D	$\frac{1}{11} = 0.09091$	[0.72727, 0.81818)
R	$\frac{2}{11} = 0.18182$	[0.81818, 1)

Figure 2.4. Table of probabilities and ranges.

With this table, the coding can begin. The encoding follows the brief algorithm listing in Figure 2.5, to encode the characters from "abracadabra" to 0.278791748370412, which is the final LOW value listed in Figure 2.6. Since there are 15 digits after the decimal place, if on average it takes $\log_2(10) \approx 3.32$ bits to represent one digit, then it takes roughly 50 bits to represent all 15 digits. The final encoding of the string is a single number between 0.1 and 1.0. Having a single number to encode the string realizes a better compression size than other entropy encoders which make a codeword for each character. However, one disadvantage is that in arithmetic coding, the entire input sequence is needed before the coding can begin, and a single corrupt bit could potentially corrupt the entire codeword.

```
LOW = 0
HIGH = 1

while(not the end of input sequence)
{
    CHAR = next character in sequence
    RANGE = HIGH - LOW
    CHAR_LOW = low value of range assigned to CHAR
    CHAR_HIGH = high value of range assigned to CHAR
    LOW = LOW + RANGE * CHAR_LOW
    HIGH = LOW + RANGE * CHAR_HIGH
}

Code = LOW
```

Figure 2.5. Arithmetic coding algorithm

	RANGE	LOW	HIGH
		0	1
A	1.00000000000000000000	0.000000000000000	0.454550000000000
B	0.45455000000000000000	0.206615702500000	0.289259090909091
R	0.08264338409090900000	0.274233020289256	0.289259090909091
A	0.01502607061983470000	0.274233020289256	0.281063120689502
C	0.00683010040024584000	0.278579447816685	0.279200366034890
A	0.00062091821820414700	0.278579447816685	0.278861686192770
D	0.00028223837608470900	0.278784712090202	0.278810370124391
A	0.00002565803418952410	0.278784712090202	0.278796374949642
B	0.00001166285944087520	0.278790013442960	0.278792133909846
R	0.00000212046688535805	0.278791748370412	0.278792133909846
A	0.00000038553943371155	**0.278791748370412**	0.278791923617362

Figure 2.6. Arithmetic coding for "abracadabra".

2.3.1.2 Huffman coding

The Huffman coding is another entropy encoder. Unlike arithmetic coding, each instance of each character in the input sequence is assigned a code. As in arithmetic coding, the algorithm begins by assigning to each character its probability of appearing in the input sequence. Subsequently, the characters enter the coding stage. In this stage, a binary tree is formed by joining the two nodes with the lowest probability to form a new node, whose value becomes the sum of its two branches. In the case where there is more than one node with the same probability, any of these nodes can be chosen. The process continues until all the nodes are part of the tree and the root node has a probability of 1.

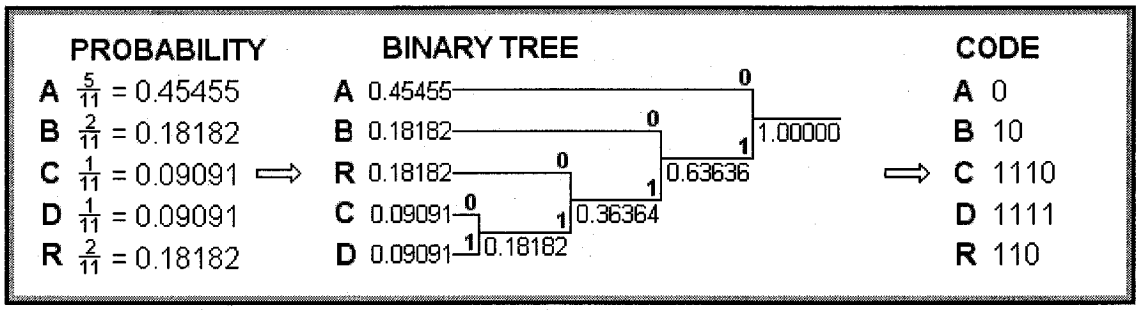

Figure 2.7. Huffman coding for "abracadabra"

In Figure 2.7, we show the Huffman coding process for the string "abracadabra". From the probability table, the binary tree is built, and from that, the codeword for each character (A, B, C, D, and R) is determined. Using the codes, "abracadabra" is encoded to 23 bits: "01011001110011110101100", which is 27 bits less than the arithmetic coder took. The compression scheme we adopted, the *bzip2* compression scheme, employs Huffman coding. The way in which the *bzip2* algorithm integrates Huffman coding in its compression, is detailed in chapter 3.

In the next section, the Ziv-Lempel algorithms are described. While it is an old compression scheme, it still remains popular. In fact, the Unix system's "compress" program employs this technique, and it is still widely used for the Graphics Interface Formats (GIF) and often the Tagged Image File Format (TIFF).

2.3.2 *Ziv-Lempel algorithms*

Ziv-Lempel (LZ) algorithms [89, 90] are dictionary methods. During compression, a dictionary is built from the components which have appeared in the past, to be used to reduce the sequence length, if that same component were to appear in the future. In other words, the compressor reads through the input data - subsequence by subsequence, constructs a dictionary of observed subsequences, and looks for redundancies as it goes. The first time a string is observed, the string is written to the output, however, any subsequent time it is encountered, a special code is then output. The output thus consists of one instance of each string, and references to these instances for the repetitions.

Ziv and Lempel developed two versions of this highly popular compression algorithm: LZ77 [89] and LZ78 [90]. The LZ77 algorithm keeps a buffer of the most recently seen data, and compares the current subsequence being encoded, with the past subsequences. The output of the compression is a sequence of triples, where the first element is the position in the buffer of where the matching subsequence starts, the second element is the length of the subsequence, and the final element is the character which follows the repeated subsequence. The LZ77 is the basis for many other LZ algorithms, for example, the LZSS by Storer and Szymanski [73], the LZFG by Fiala and Greene [23], and the LZRW by Williams [82]. Moreover, further improvements were made in later LZ algorithms [4, 32].

Unlike the LZ77 algorithm which works on past data, the LZ78 algorithm actually works on future data. This is done by forward-scanning the input buffer and matching it against values in a dictionary it maintains. Like the LZ77, the LZ78 also gave rise to other versions [34, 56, 81].

2.3.3 *JPEG compression*

The JPEG compression algorithms are a family of compression techniques for images standardized by the Joint Photographic Experts Group [51]. The block diagram for the JPEG compression technique is shown in Figure 2.8. The Forward Discrete

Cosine Transform (FDCT), or (DCT), is an algorithm that partitions the image into 8 x 8 pixel blocks, and transforms these blocks into the spatial frequency. At this point, none of the data has been lost; it is the quantization stage which is lossy. In this stage, the DCT coefficients are divided by their corresponding quantization coefficients and rounded to the nearest integer.

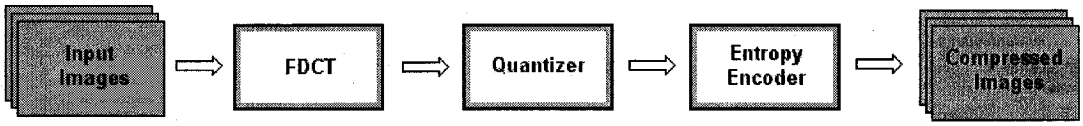

Figure 2.8. Block diagram for JPEG compression techniques.

The quantization coefficients are not fixed, but rather, can vary with the desired quality level (Q factor). Lower Q factors produce better results; however, the closer to the original image, the larger the size of the compressed image, and thus the less effective a compressor.

2.3.4 Other universal compression algorithms

Since the Ziv-Lempel algorithms, other more recent compression algorithms have been introduced. The partial matching algorithm (PPM), the dynamic Markov coding (DMC) algorithm, and the switching method, are just a few of the many compression schemes available which follow the modeling-coding paradigm.

The PPM algorithm [16], developed by Cleary and Witten, predicts the probability of occurrence of the current symbol by using the frequencies of all the characters' occurrences in the past. This probability is then used to encode the symbol with the arithmetic coder described in Section 2.2.1.1. In addition to Cleary and Witten's algorithm, other PPM algorithms were developed by Bunton [8-10] and Shkarin [69].

In the late 1980s, Cormack and Horspool developed the DMC algorithm [18, 35]. The DMC attempts to determine which Markov source has produced the input

sequence that is to be compressed. A Markov source is a finite state machine (FSM) with a set of states and transitions. There can be, at most, k transitions from each state, and all of them are denoted with different symbols. At each transition, there is a certain probability of being chosen for each state. Further details of the algorithm can be found in [18], and a detailed description of the generalized dynamic Markov coder (GDMC), developed by Teuhola and Raita, can be found in [74].

The switching method, proposed by Volf and Willems [78], employs two compression schemes (PPM, LZ77, DMC etc) in combination. In order to obtain the best compression ratio, the switching algorithm decides which part of the input sequence is to be compressed with which of the two compression schemes chosen.

In our anomaly detection algorithm, we employed the *bzip2* compression scheme, which is derived from the Burrows-Wheeler Transform (BWT) [11], a block-sorting lossless algorithm. The BWT processes the input file in blocks. A single block is read, compressed, and written to the output before the next block is processed. The benefits in using a block-sorting algorithm lies in its ability to find redundancies within a block of data, thus, in our application, since we are looking for redundancies between two images, comparing them in blocks is preferred. The details of the BWT, along with a justification for using it, are presented in the following chapter.

CHAPTER 3

ANOMALY DETECTION ON INTENSITY IMAGES

The anomaly detection that we implemented is a compression-based similarity metric based on the normalized compression distance (NCD) proposed by Li *et al* [48]. They proposed that the natural measure of the information content of a data file, is given by the smallest size to which the file can be compressed. Compression programs are designed to minimize the amount of storage resources required by a given file, by removing any redundancies within the file, in order to produce a much smaller file from which the original can still be reconstructed. Thus, given that two similar files will have more redundancies between them than two dissimilar files, a compression program can be used to determine the level of (dis)similarity between two files. Li *et al* proposed a compression distance which compares the compression size of the concatenation of two files with the sum of the compression sizes of the individual files.

$$NCD(x,y) = C(xy) - \min\{C(x), C(y)\} \tag{3.1}$$

Unfortunately, the compression distance is skewed by the size of the files being compared. For example, though intuition says that two larger files with only a small fraction of redundancy is not as dissimilar as two smaller files with the same amount of redundancy. Without normalization, however, the compression distance would deem the two larger files as dissimilar as the two smaller ones. Thus, Li *et al* proposed a normalized compression distance (NCD):

$$NCD(x,y) = C(xy) - \min\{C(x), C(y)\} \tag{3.2}$$

Our similarity metric is based on the NCD, however, as in Bennett *et al*'s technique [5], we compare the size of the compression of the concatenation of the two files with the sum of the individual files' compression sizes:

$$p(I_1, I_2) = \frac{size(C[I_1]) + size(C[I_2]) - size(C[I_1 \oplus I_2])}{size(C[I_1]) + size(C[I_2])} \quad (3.3)$$

where C[] indicates the compression operation, and ⊕ indicates the concatenation of the data sets. If I_1 and I_2 are completely different, then the concatenation of the two files should be almost as big, if not as big as the two individual compressed files, and thus, the similarity measure should be 0, or close to 0. If I_1 and I_2 are identical, then an ideal compressor would yield a similarity measure of 0.5. In other words, files with greater similarity have a higher similarity measure than those with greater dissimilarity, and the range of the similarity measure is [0, 0.5].

The concatenation is performed using the UNIX command *cat,* where given two image files I_1 and I_2, *cat*(I_1, I_2), would produce one continuous string, I_{12}, which is simply I_2 appended to I_1 [15].

3.1 *bzip2* compression scheme

Since the anomaly detection is based on a compression-based technique which seeks to find (dis)similarities between images, an algorithm which compresses in blocks is required. One such algorithm is the *bzip2* algorithm [29], which is derived from the Burrows-Wheeler transform (BWT) [11]. The BWT applies a reversible transformation to the data, after which, the data undergoes a move-to-front (MTF) [11] encoding. Finally a Huffman encoder [37] is applied to the data.

3.1.1 The Burrows-Wheeler Transform

The BWT does not process the data sequentially, but rather, accepts as input blocks of data in 100 to 900 KB block-sizes. The transform works by taking a string S of length N, and forming an N-by-N rotation matrix M by performing N left cyclic shifts. In other words, all the characters of the string are shifted left by one position, and the first character is moved to the end of the string until N strings are formed. The rows of the M matrix are then sorted lexicographically (or alphabetically) to form M', and its last column forms the string L of length N. The purpose of the transformation is

to group the characters in such a way so that redundancies can be found. After the transformation, the string L, as well as a primary index, I, which indicates the position in the string L of the original first character of S are stored.

To illustrate how the BWT works, we now present an example where the string "`wheeler`" is compared to itself. Thus, we would like to find the similarity metric between a string and itself.

$$\rho(\text{"wheeler"},\text{"wheeler"}) = \frac{size(C[\text{"wheeler"}]) + size(C[\text{"wheeler"}]) - size(C[\text{"wheelerwheeler"}])}{size(C[\text{"wheeler"}] + size(C[\text{"wheeler"}])}) \quad (3.4)$$

As shown in the above equation, the largest string that will be compressed is the concatenation of "`wheeler`" to itself. In Figure 3.1 the BWT is applied to "`wheelerwheeler`". In this example, we assume that the block-size is at least 14 bytes, thus, the entire string can be compressed in one block. After cyclically shifting and lexicographically sorting, L becomes "`helhelwweeeerr`" and the primary index, I is 6, when the index begins at 0. With this primary index, the transformation is reversible.

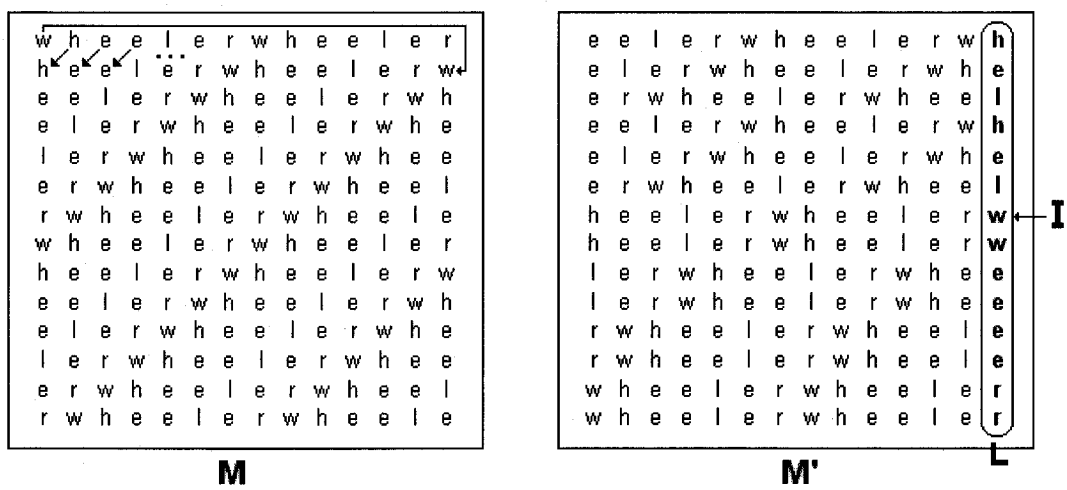

Figure 3.1: BWT applied to "`wheelerwheeler`".

3.1.2 *Move-To-Front coding*

The string L is then coded, using move-to-front (MTF) coding, to form the vector R, where each element in R is a code for each character in L. The code is derived by assigning to $R[i]$, where $i = 1 \ldots N\text{-}1$, the number of characters preceding $L[i]$ in the vector Y, which is initially made up of one instance of each character in L sorted lexicographically. In Figure 3.2, the MTF coding is performed on the output of the transformation of "`wheelerwheeler`". The MTF coder is given the strings L and Y as input. $R[0]$ is assigned the value of 1 since there is 1 character preceding $L[0]$ in the input Y, "`ehlrw`". A new Y is then formed by moving the 'h' to the front of the string, thus forming "`helrw`". $R[1]$ is then determined for the character $L[1]$, and so on, until the N-length code is determined to be "`11222240200040`".

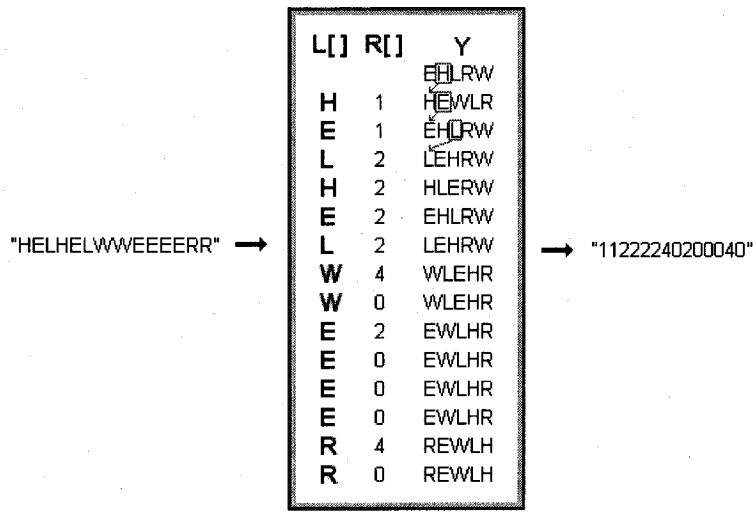

Figure 3.2. MTF for "`wheelerwheeler`" with 14 byte block-size.

3.1.3 *Huffman coding*

The final step for compression is to apply Huffman encoder to the vector R. The probability of each element in R is determined. In Figure 3.3, the Huffman encoder is applied to the output R code from the example in Figure 3.2. After building the tree, based on the probabilities for each of the characters, each character in the R string is encoded. For example, $R[0]$=1 is in the third level of the tree, and is reached by

following the branches 110. In other words, it took 3 characters to encode the character '1'. Moreover, the characters '0', '2', and '4', take 1, 2, and 3 characters to encode respectively. Thus, the total number of characters necessary to encode the *R* string is 27. In other words, the string "`wheelerwheeler`" was compressed to 27 bits.

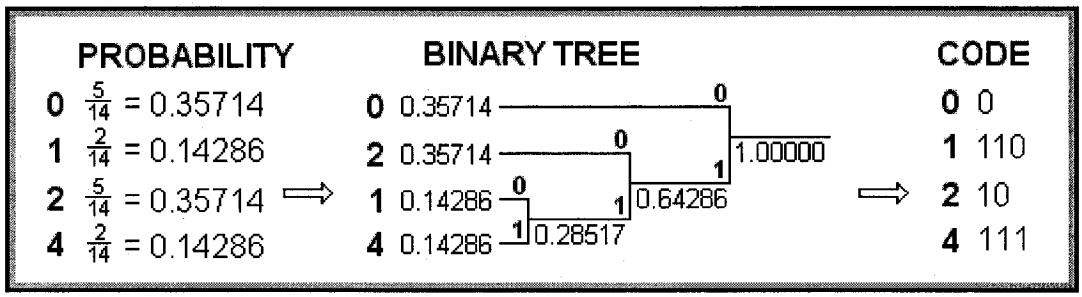

Figure 3.3. Huffman coding for "wheelerwheeler" with 14 byte block-size.

By default, the *bzip2* compressor compresses files in 900 KB block-sizes. In other words, if the size of the file to be compressed is greater than 900 KB, then the file is split into 900 KB blocks before being compressed. Because the BWT works by removing redundancies, in order for the metric not to be skewed, the largest file compressed must be less than 900 KB. The largest file compressed with the similarity metric shown in Equation 3.3, is the concatenation of the two files being compared. Thus, the sum of the sizes of the files to be compared must be at most 900 KB. In the previous example, the "`wheelerwheeler`" string was compressed assuming that the block-size was large enough that the string did not need to be split. To demonstrate that the metric is skewed, if the string needs to be split before compression, the example is repeated, however this time with a 9-byte block-size. Thus, the string "`wheelerwheeler`" must be compressed in two blocks: "`wheelerwh`" and "`eeler`". From Figure 3.4, it can already be seen that by splitting the string, some of the redundancies that was achieved with the 14-byte block-size, is not achieved here.

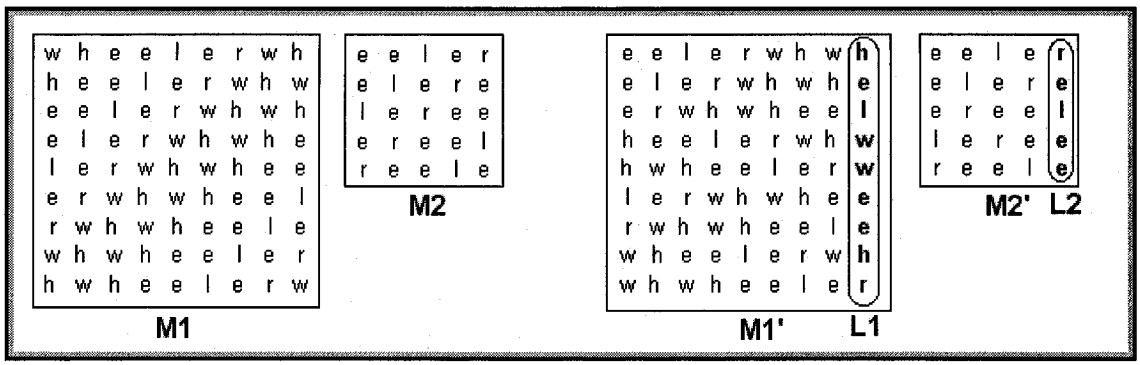

Figure 3.4. BWT for "wheelerwheeler" with 9-byte block-size

From the 9-byte block-size example, $L1$ and $L2$ are found to be "`helwweehr`" and "`relee`" respectively. The L strings are encoded using MTF coding and then Huffman coding. Since there are two strings being compressed, each string requires its own binary tree for the Huffman coding. The final encoded string is "`010110110001000111110`" for $L1$ and "`10010011`" for $L2$, which are 21 and 8 bits respectively. Thus, the string "`wheelerwheeler`" was compressed to 29 bits when using a 9-byte block-size, which is 2 more bits than when using a block-size that did not require splitting the string.

Now, we consider the effects of splitting the string on the similarity metric. The string "`wheeler`" is compressed following the same process that we used. Because it is a 7-byte string, both the 9-byte and 14-byte block-sizes compressed "`wheeler`" to "`001011010111110`", which is 15 bits. Plugging this value, along with the previously computed compression sizes into the similarity metric equation, yields:

$$\rho_{14-byte} = \frac{15+15-27}{15+15} = 0.1$$

$$\rho_{9-byte} = \frac{15+15-29}{15+15} = 0.3\overline{3}$$

Thus, the similarity measure between "`wheeler`" and itself, was found to be more dissimilar when using a block-size that required the string to be split before

compression. In our implementation, we used the default block-size of 900 KB. Thus, the sum of the sizes of the two images being compared must be less than 900 KB. The images from our demonstration system are all roughly 102 KB, thus the concatenation of the two images are roughly 204 KB and far below the maximum size of 900 KB.

In the next section, we present a brief listing of our algorithm for the anomaly detection. Subsequently, we present the results of our demonstration system applied on intensity images.

3.2 Our Anomaly Detection Algorithm

Figure 3.5 shows the basic algorithm followed for the anomaly detection. When the anomaly detection algorithm begins, an image is downloaded from the camera. As explained in Chapter 1, the images acquired from the camera are in JPEG format. Although this format is inherently its own compression scheme, we cannot compare the JPEG image sizes. With JPEG compression, the images are manipulated in 8 x 8 pixel block-sizes; however, we require a compression scheme that can find redundancies in concatenated images. Thus, the input image, I_0, is converted into bitmap (BMP) format and then compressed using the *bzip2* compression algorithm. If the input image is the first image to be processed, it is marked and saved as a novel image. Otherwise, I_0 is compared to all the images previously deemed novel by compressing the previously stored image, I_1, and concatenating I_0 and I_1 to form I_{01}. Using the similarity metric defined in Equation 3.3, the level of similarity between I_0 and I_1 is determined and compared against an empirically chosen threshold. If the two images have a similarity measure below the threshold, they are deemed dissimilar; conversely, if they have a similarity measure above the threshold, they are deemed similar. In the case of the former, the input image is then compared to the next image from the novel image list until it has been compared to all the images on the list, or it is deemed similar to one of the images on that list. If the image is dissimilar to all of the images on the list, then it is considered novel and it is added to the novel image list. The compression of the image is then saved. If the image is similar to one of the images on the list, then the

image can be discarded, since only one instance of the similar images is required for future comparisons. In other words, the input image is only saved and compared against new incoming images if it is deemed novel. Thus, rather than storing every image processed by the system, by storing only the novel images, only the images with useful information are kept, and storage resource requirements are reduced. Moreover, when only the novel images are stored, computational resources are reduced as well, since fewer comparisons are required for each incoming image.

```
1:  Download an input image from the camera
2:  Assign the input image to I0
3:  Convert I0 to BMP format from JPG format
4:  C[I0] = Compress I0 using bzip2
5:  if I0 == first image processed
6:        novel_image_list.add(I0)
7:        Save C[I0]
8:  else
9:        while(!eof(novel_image_list))
10:             I1 = data from current position of novel_image_list
11:             C[I1] = Compress I1
12:             I01 = concatenate(I0, I1)
13:             C[I01] = Compress I0114:
      p(I1,I2)=(size(C[I1])+size(C[I2])-size(C[I1⊕I2]))/(size(C[I1])+size(C[I2])), // ρ = similarity
15:             if ρ<threshold (τ)              // dissimilar
16:                   go to step 10
17:             else                            // similar
19:                   increment I1.match_count
18:                   Sort novel_image_list by match_count
19:                   break from loop;
20: if I0 dissimilar to all images in novel_image_list
21:       novel_image_list.add(I0)
22:       C[I0] = Compress I0
23: go to step 1.
```

Figure 3.5 Pseudocode for anomaly detection algorithm.

Since the input image is compared to the images from the novel image list until a similar image is found, it is clear that finding a match sooner reduces the number of comparisons required and thus, reduces the computational load. Thus, the order in which the comparisons are made is pivotal. Step 18 of the algorithm listed in Figure 3.5 shows that the novel image list is sorted by the amount of times that they have been deemed similar to the input, by the most often to the least. The reasoning is that an input that will ultimately be determined to not be novel, will most likely be considered

similar to one of the images that have been seen most frequently in the past. For example, if the video camera were surveying a corridor, then a likely assumption would be that an empty corridor would be the most frequently viewed scenario. Thus, it would be advantageous to rule out this image first by comparing the input image to it first.

This approach to anomaly detection exploits geometric structure implicitly, in that the comparisons are based on the image itself. Rather than describing each image according to its geometric shapes, and comparing the descriptions, the entire images are compared. While using the geometric structures in a more explicit way could provide a more precise definition of similarity, the tradeoff is between speed and precision. In this way, we achieve a simple method for anomaly detection.

3.3 Results

Before the algorithm could be properly tested, an appropriate threshold needed to be chosen for the similarity measure. The anomaly detection algorithm was run on a test set of 100 images, and then, the similarity measure was empirically chosen by manually sorting through all the images, deciding which pairs of images should be considered similar, and/or, which should be considered dissimilar. Although some manual training was required here, the training time is minimal. Moreover, some user input of what is considered similar is beneficial, since it depends on the user's perspective. What was noticed during this classification process was that for some pairs of images that intuition would deem similar, the similarity measure would be lower for these pairs of images than for a pair of images that intuition would deem dissimilar. Moreover, for certain pairs of images that intuition would deem dissimilar, the similarity measure would be higher for these pairs of images than for pairs of images that intuition would deem similar. The difficulty in choosing a threshold thus lies in determining where to place the threshold, so that the greatest amount of dissimilar images were found without triggering too many false positives; which are similar images being marked as novel. In other words, when choosing a threshold, there is a tradeoff between false positives and false negatives. Ultimately, because the anomaly detection algorithm is for video

surveillance applications, we decided that it was more important to catch all the anomalies while risking classifying too many images as dissimilar, than to risk missing any anomalies. With that in mind, a conservative threshold of 0.055 was chosen for the test.

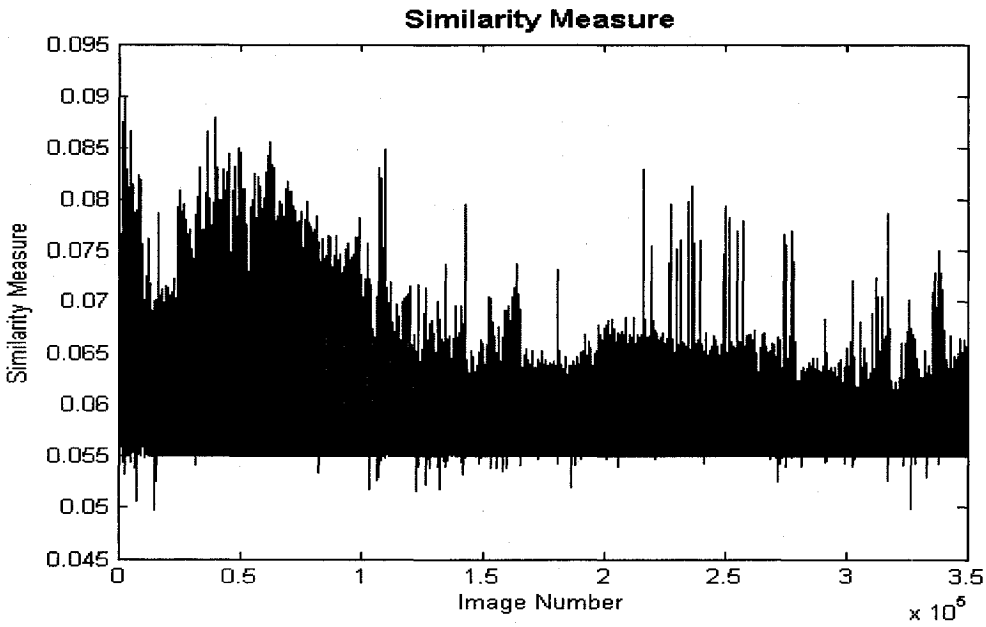

Figure 3.6. Highest similarity measure for each intensity image processed over 2 days

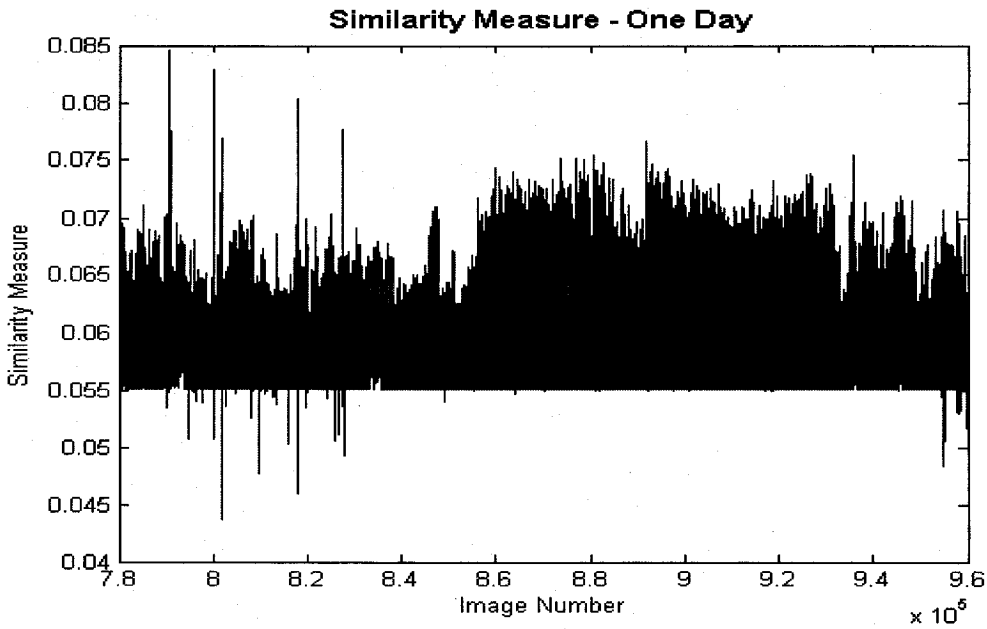

Figure 3.7. Highest similarity measure for each intensity image processed over 1 day

The anomaly detection algorithm was run over a six week period. In Figure 3.6, the highest similarity measure for each of the first 2 days of the test was plotted, which is 350,000 images. The points below 0.055 are the highest similarity measures for novel images. The plot shows that there is only the occasional sharp dip, meaning that novelty is rare. Figure 3.7 shows the plot for just one day of activity. Here, we notice that the majority of the novel images are found in the first quarter of the plot, which corresponds to the first 6 hours of the work day.

In Figure 3.8, the number of novel images detected was plotted over the total number of intensity images processed over the six week period. In total, over 6 million images were acquired from the demonstration system. Of these images, 967 novel images were detected. In the first week, nearly 400 novel images were detected; however, after another 5 weeks, the number of novel images only increased by fewer than 600 images. Therefore, as the cubic curve in Figure 3.8 shows, the rate of novelty is decreasing with time, which is not surprising, since we expect that as with a security guard, initially, the system learns a lot; however, over time, it establishes a baseline of normal behaviour.

The plot in Figure 3.8 also shows several steep rises followed by plateaus. We notice that the beginning of the steep rises correspond to the beginning of a new work day, whereas, the flat lines correspond to the inactivity overnight. In fact, in Figure 3.9, we zoomed in on the events of 16/05/05 from Figure 3.8. The plot shows that the majority of the activity occurs in the morning, starting just before 9 AM. Over the day, not only does much of the activity taper off, but the activity that does occur is often similar to an event captured earlier in the day. For example, if a person were to walk down the hallway in the morning, during the course of the day, if the person were to walk down the hallway several times, the system would not deem these events as novel, since it has already captured images of a similar event in the morning.

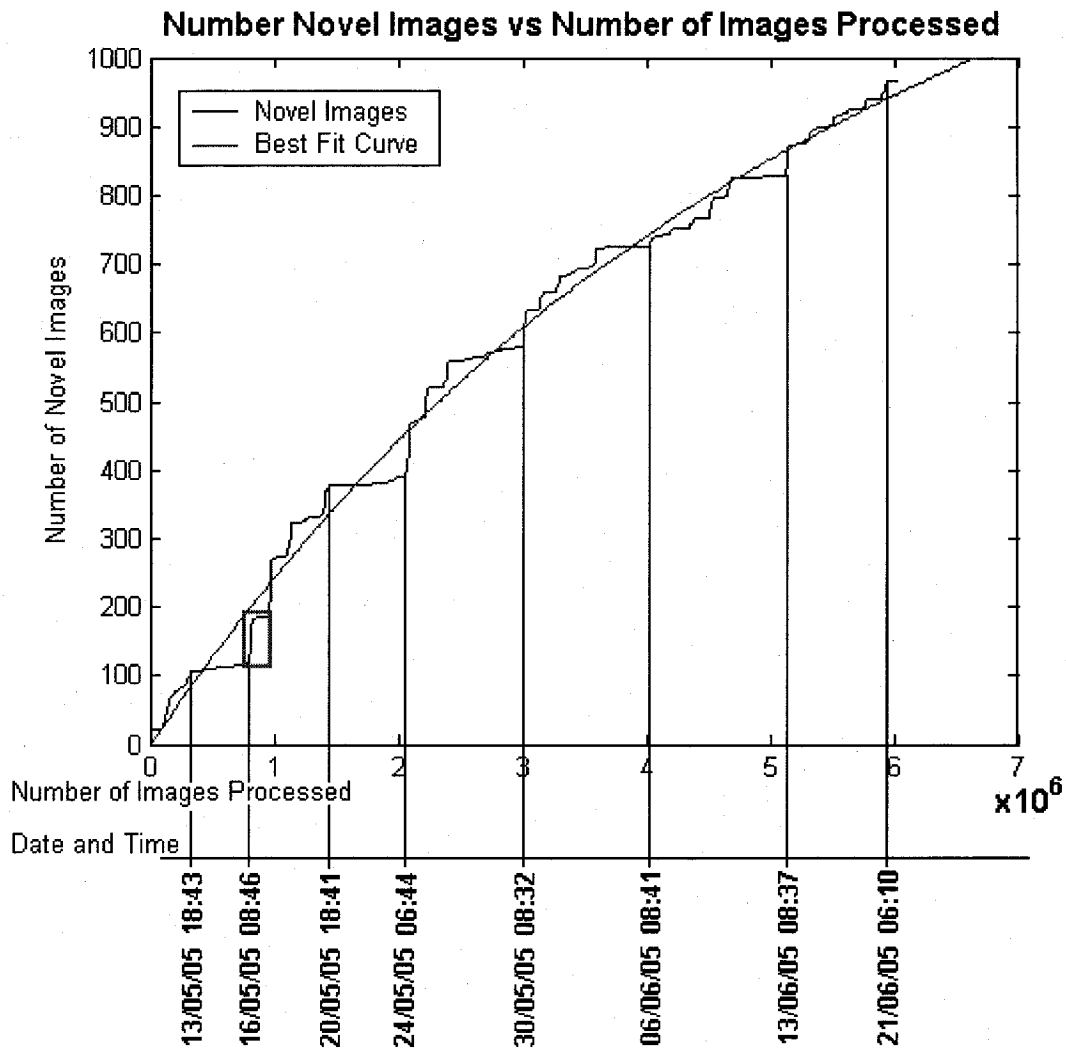

Figure 3.8. Number of novel images vs. number of intensity images processed over 6 weeks.

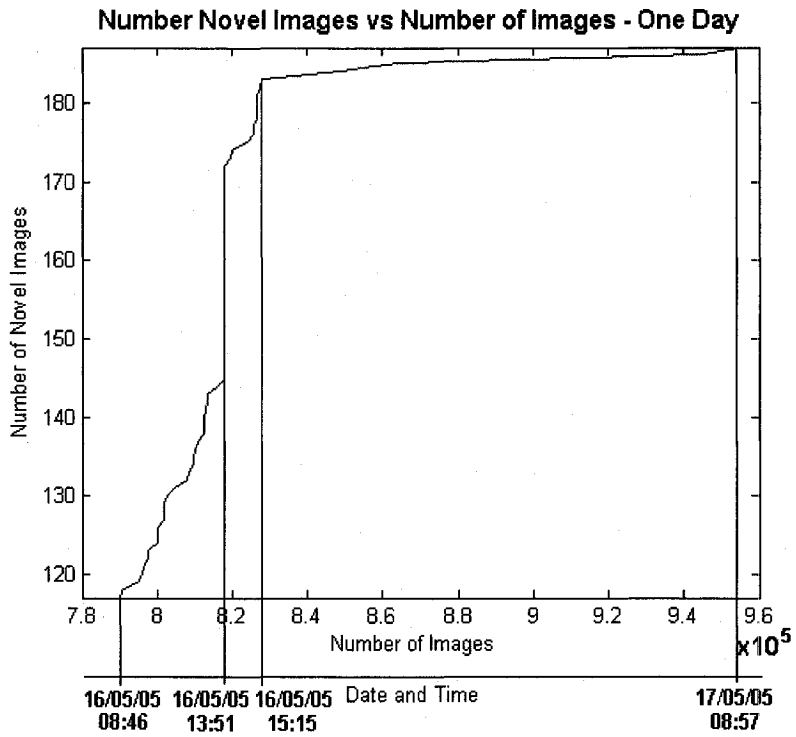

Figure 3.9. Number of novel images vs. number of intensity images processed over 1 day.

Referring again to Figure 3.8, we note that during the weekend, (13/05/05 – 15/05/05) there was barely any activity in the research center, and thus, very few novel images were detected. The lack of activity because of the weekend can be further confirmed by the following weekend's (20/05/05 – 23/05/05) inactivity. The second weekend was actually a long weekend, since Monday was a statutory holiday. We note that the lack of activity also carried through that Monday holiday. The steep ascent of novel images only begins the following day, on the Tuesday (24/05/05).

In Figure 3.10, examples of similar and dissimilar image classification are shown. When compared, the top two images were deemed similar, thus only one instance of the two images was added to the novel image list. However, the bottom image was deemed dissimilar from either of the top two images, and thus, it was added to the novel image list. It is difficult to provide an intuitive description of what makes images (dis)similar. By observing the example images though, we see that the people in the similar images have similar grey level clothing, and are a similar size in the image,

whereas, the person in the bottom image is donning different clothing and is smaller than the people in the similar images.

Out of the 6,032,556 images examined, 55 were classified as novel when intuition would say that they were not novel, which is a 9.1×10^{-6} false positive rate. The majority of these false classifications can be attributed to the anomaly detector's sensitivity to illumination changes. For example, in Figure 3.11, the two images were both saved as novel images; however, with the exception of the difference in illumination, one could say that the two images are similar. The advantage to our system is that regardless of the illumination changes, the system adapts, since it learns online. Thus, the wrongly classified novel images are simply added to the list of novel images, and any subsequent input images that resemble this image will not be novel. However, in spite of the adaptability of the system, the algorithm's sensitivity to illumination is still an issue, since false positives trigger false alarms which can be costly in real-world applications. We will thus address the issue of false positives in chapter 4.

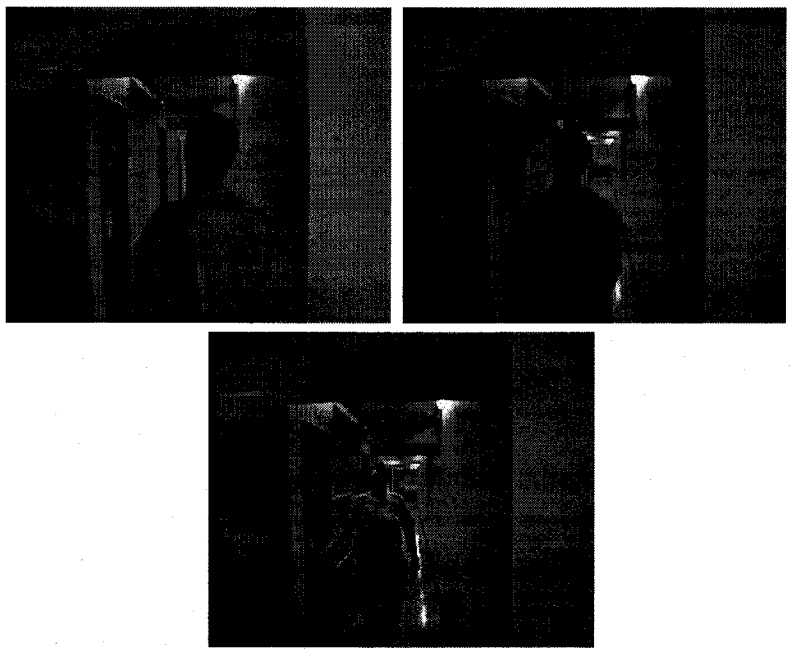

Figure 3.10. Examples of similar (top) and dissimilar (bottom) image classification.

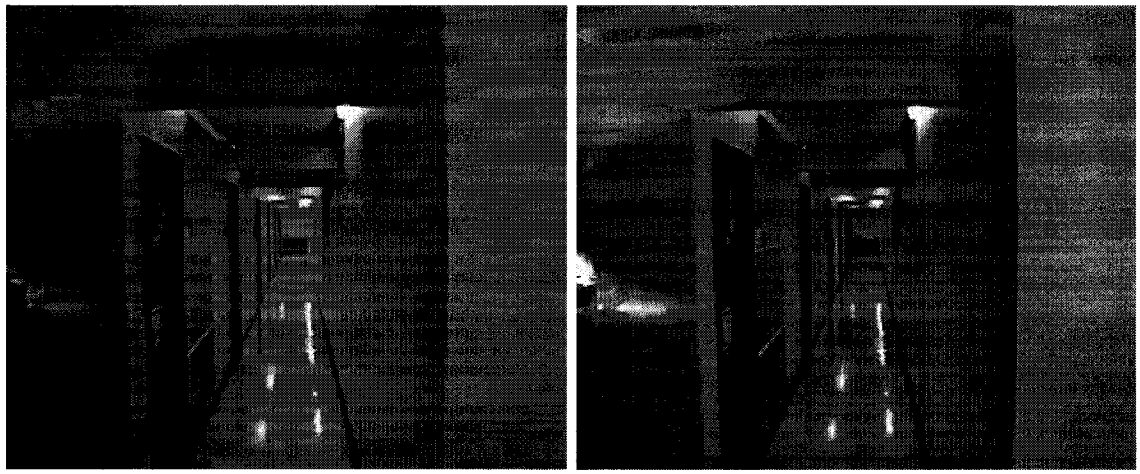

Figure 3.11. Example of a false positive: two similar images classified as dissimilar.

Because of the difficulty of intuitively describing similarity, the exact description for false negatives is difficult to calculate; of course, since it is highly dependent on what we consider similar. What we did notice, however, was that at this high-level of comparison, images containing two different people with similar clothing and similar appearances could be considered similar. Thus, in chapter 5, we crop out the regions of the images containing people, and apply the anomaly detector on these images.

In spite of sorting the novel images in order of likelihood to be matched to the input image, the number of novel images continues to grow, implying that the amount of storage resource required will also continue to increase with time. As aforementioned, each image is roughly 102 KB. Which means that at 967 novel images, approximately 96 MB of the hard disk is required. Now, the intensity images compress to anywhere between 18-21KB. Thus, by storing the compressed version instead, the storage resources required have been reduced by around 80%. After around 6 weeks of video surveillance, only 19 MB of images are stored. In the following two chapters, the issue of storage resources is further addressed.

In order to act as an effective video surveillance system, the time the system requires to ascertain the novelty of each frame is very important. To simulate a realistic

real-world situation, the anomaly detection was performed on a Unix machine which had other processes running in the background. Although assigning a high priority to our process might ensure that it would receive most of the quanta (time slices assigned to processes) on the Central Processing Unit (CPU), and thereby improving the processing time of each frame, our system was not assigned a high priority, because a dedicated machine may not always be available in real-world applications.

A stripped-down version of the anomaly detection algorithm was run for 24 hours, where 157993 images were processed. We found that, on average, each frame takes 0.547 seconds to process. Given that the maximum frame rate of the netcam is 2 frames/sec, as specified in Chapter 1, a frame rate of 1.829 frames/sec is not unreasonable. We recall that the anomaly detector works by comparing the input images to the previously-seen novel images until either a match is found, or the input has been compared to all of the stored images. Since the novel image list is sorted by the likelihood that the input will be similar to it, the processing time of images that are eventually deemed similar, does not increase much with time. The majority of the time, a match will be found within the first two comparisons, and thus, the majority of the time, a match is made in less than a second. However, in the case of the novel images, the processing time is directly proportional to the number of novel images stored. In Figure 3.12 we see that the processing time for novel images grows linearly with the number of novel images. At around 170 novel images, the processing time is 9 seconds/frame. In Figure 3.13, the processing time for each novel image is plotted against the total number of images processed. The 9 second worst-case time occurs near the 160,000 image. In chapter 4, we will discuss how, by first applying an edge detector on the images, the speed of the algorithm is improved.

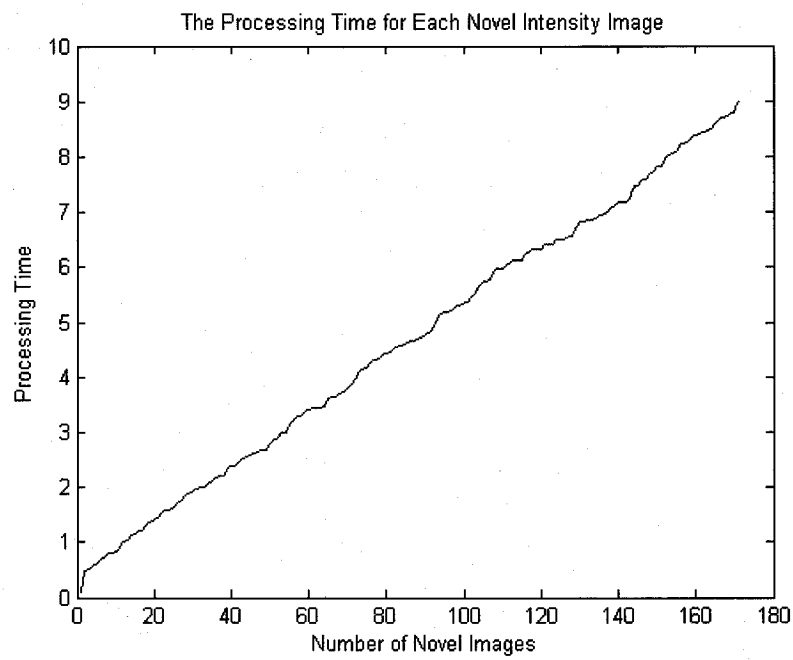

Figure 3.12. Processing time for each novel intensity image.

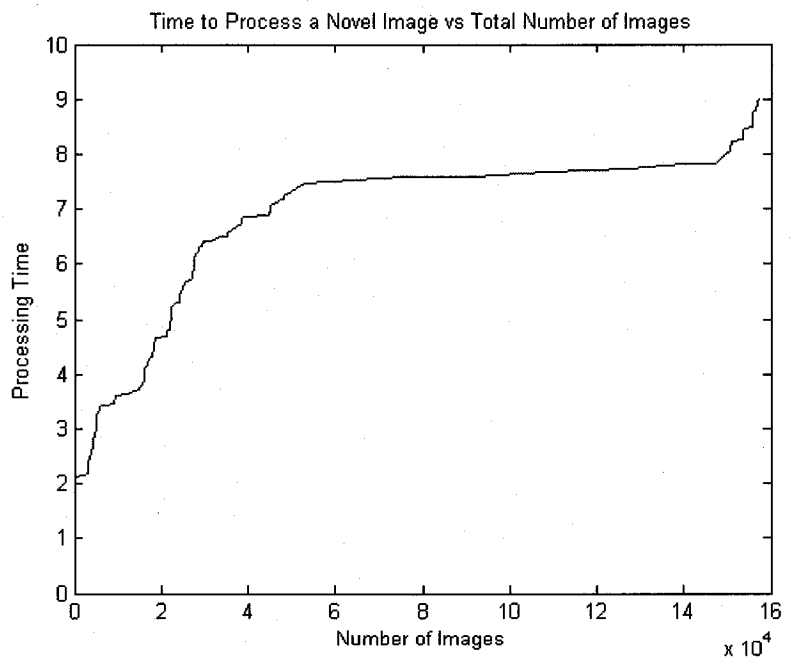

Figure 3.13. Processing time for each novel image vs. total number of images processed.

CHAPTER 4

ANOMALY DETECTION ON EDGE IMAGES

As discussed in Chapter 3, the major disadvantage to applying the anomaly detector to the intensity images was that the algorithm was sensitive to changes in illumination. In fact, the intensity images contained information regarding illumination changes and other information that could, otherwise, be discarded to speed up computation and also save on storage resources. In computer vision, an edge detector removes this information while maintaining the structural properties of the image. As we would like to reduce or remove the algorithm's sensitivity to illumination changes, we thus apply an edge detector on the images before applying the anomaly detector.

In this chapter, we begin by presenting the edge detector we applied and how it is integrated with our anomaly detection algorithm. Subsequently, we present the results of the tests which we ran on our demonstration system, with the edge detector in place. And finally, we compare the results of the edge detector in terms of accuracy, storage resources and speed.

4.1 Integrating the Canny Edge Detector

The Canny edge detector is applied to the images before being compressed and checked for novelty. In Figure 4.1 the anomaly detection algorithm with the Canny edge detection is listed. Essentially, the algorithm remains the same as in Section 3, however, in this case, an edge detector is applied to the images before they are processed (see step 4).

```
1:  Download an input image from the camera
2:  Assign the input image to I0
3:  Convert I0 to BMP format from JPG format
4:  I0 = edge detector applied to I0
5:  C[I0] = Compress I0
6:  if I0 == first image processed
7:        novel_image_list.add(I0)
8:        Save C[I0]
9:  else
10: while(!eof(novel_image_list))
11:             I1 = data from current position of novel_image_list
12:             C[I1] = Compress I1
13:             I01 = concatenate(I0, I1)
14:             C[I01] = Compress I01
15:             p(I1,I2)=(size(C[I1])+size(C[I2])-size(C[I1⊕I2])) / (size(C[I1])+size(C[I2]))   // ρ = similarity
16:       if ρ<threshold (τ)                    // dissimilar
17:                 go to step 10
18:       else                                  // similar
19:             increment I1.match_count
19:             Sort novel_image_list
20:                 break from loop;
21: if I0 dissimilar to all images in novel_image_list
22:       novel_image_list.add(I0)
23:       C[I0] = Compress I0
24: go to step 1.
```

Figure 4.1. Pseudocode for anomaly detection on edge images

In Figure 4.2, the resulting edge image of a given intensity image is shown. The edge image is a binary image with the edges represented by white pixels.

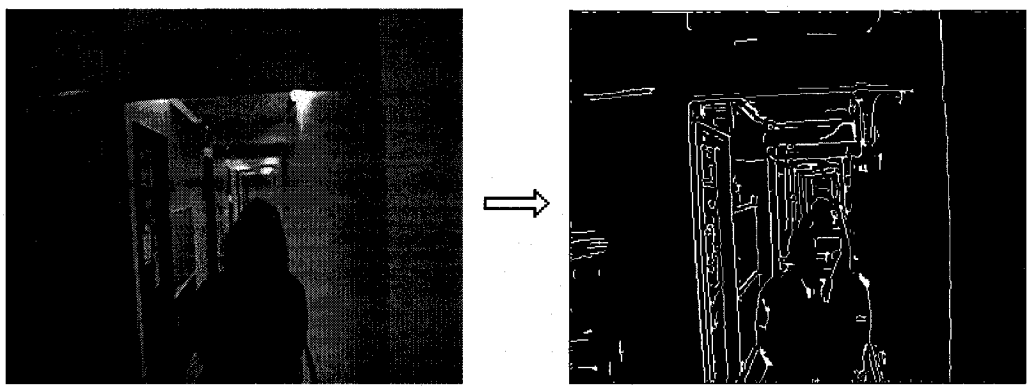

Figure 4.2. Example of an intensity image and its corresponding edge image.

In the following section, we will present the results of the test which we ran on the anomaly detection, on edge images acquired from the camera. Subsequently, we

will compare the results of the edge images to the intensity images. In particular, we will address the issue of the algorithm's sensitivity to illumination changes and how the edge images eliminate this issue.

4.2 Results

We ran the anomaly detector on our demonstration system over 5 days. After applying an edge detector, only a small portion of the image pixels become relevant, the remainder of the pixels is set to black. We can see from the example image, in Figure 4.2, that the majority of the edge image is black, and thus, a higher level of similarity is found between two edge images, and a larger threshold is required. We found that a threshold of 0.17 was a good threshold. In Figure 4.3, the highest similarity measure for each edge image was plotted. The spike which occurs near image 2.8×10^5, is the result of the camera going temporarily offline. The algorithm detected and noted the anomaly, and then recovered. In real-world applications, we would expect the camera to occasionally go offline for any number of reasons, thus, it is imperative that the system can recover from such occurrences. After 8000 images, we can see from Figure 4.3 that the band of similarity measures shift down. This shift is likely to be caused by the camera being moved. In chapter 5, we will address the issue of the algorithm's sensitivity to slight changes in the camera position. The occasional sharp dips again confirm that novelty is rare.

From Figure 4.4, we see that out of 567,840 images processed, 420 novel images were found. Though far more novel images were found after 5 days, in this case, than when the anomaly detector was applied to intensity images for 5 days (see Figure 3.8), we cannot use this as an indication that the algorithm find more novel images with edge images. Several factors affect the number of novel images found. Firstly, the threshold chosen affects the results. A threshold of 0.17 may have been more conservative than the threshold of 0.055 for the intensity images. Moreover, there may have simply been more activity in the corridor on the days where the anomaly detector was applied to the edge images.

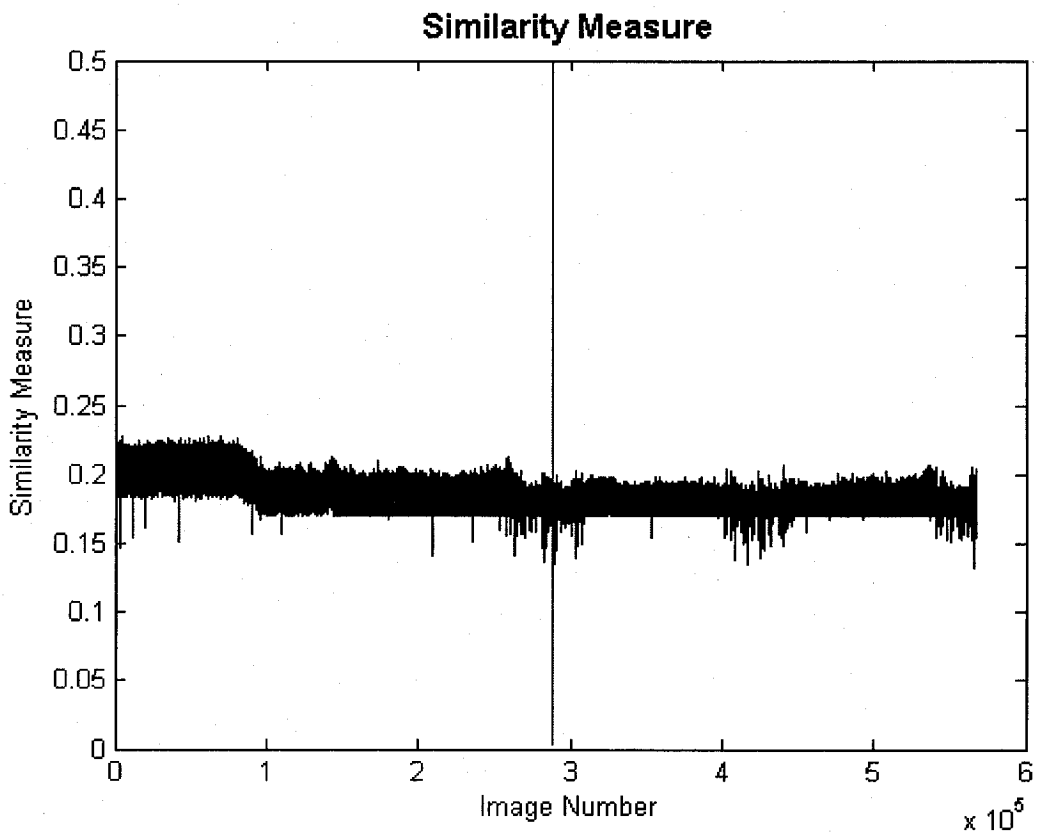

Figure 4.3. Highest similarity measure for each edge image.

In Figure 4.5 and Figure 4.6, we present the plots of another test, where the anomaly detector was applied to the same set of images for both the intensity and the edge images. In this test, we see that there were more intensity than edge novel images found. Here, we see that the distinction between the highest similarity measure for similar and dissimilar images is much more clearly defined for edge images than for intensity images.

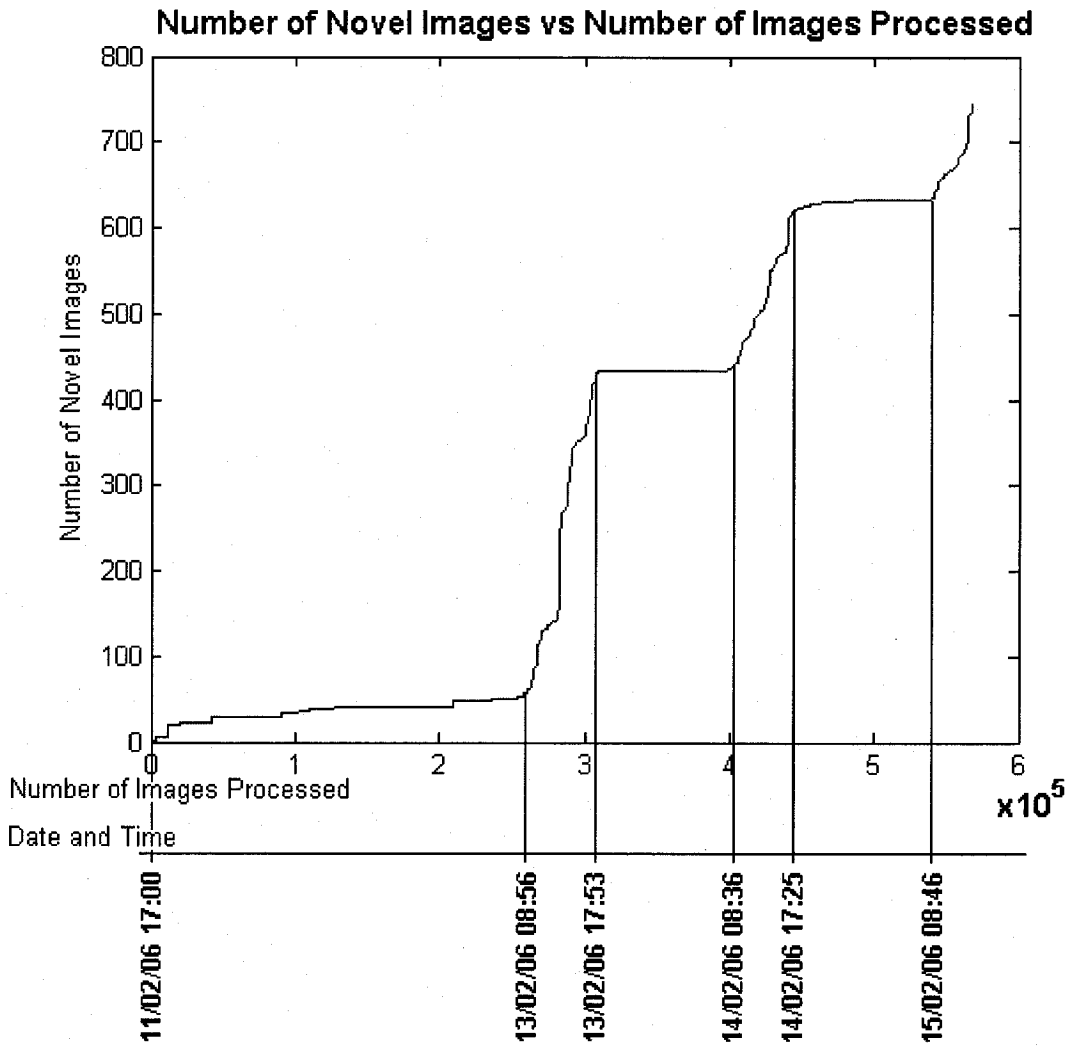

Figure 4.4. Number of novel images vs. number of edge images processed over 5 days.

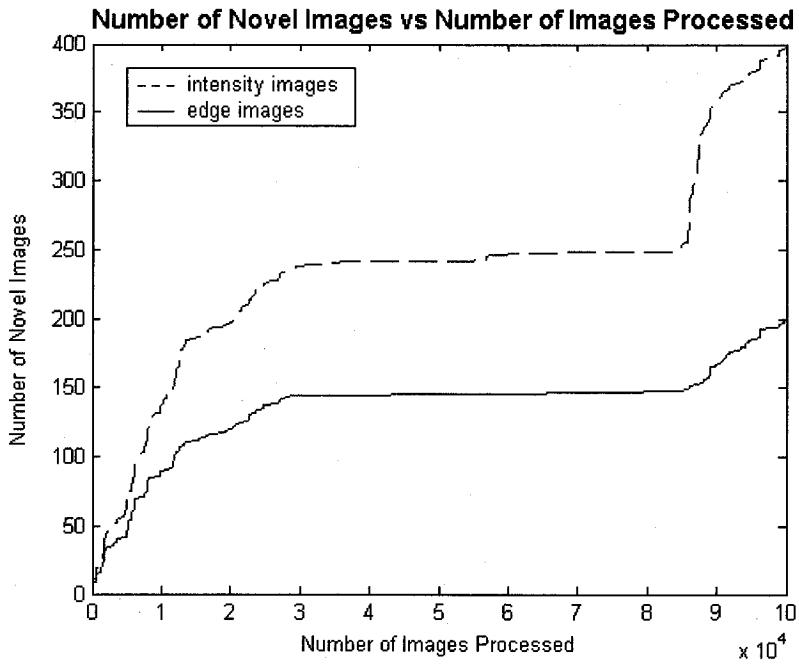

Figure 4.5. Number of Novel Intensity and Edge images vs Number of Images Processed

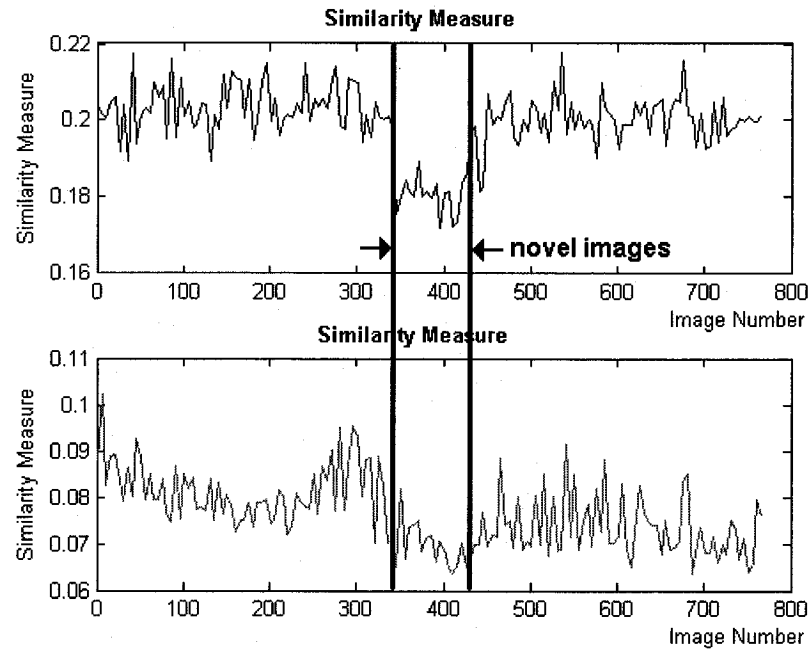

Figure 4.6. Highest similarity measure for intensity and edge images

In chapter 3, we presented the issue of sensitivity to illumination changes. For the most part, applying an edge detector removed this sensitivity. In Figure 4.7, the example images from Figure 3.11 are shown after having an edge detector applied to them. It can be observed that most of the illumination information was removed during the edge detection process. Thus, when the two images were compared, a great deal of similarity was found between them, and the anomaly detector correctly deemed these two images as similar. Because of a measure of illumination, invariance was achieved with the edge images; the number of false positives due to illumination changes that occurred with the intensity images, did not occur here.

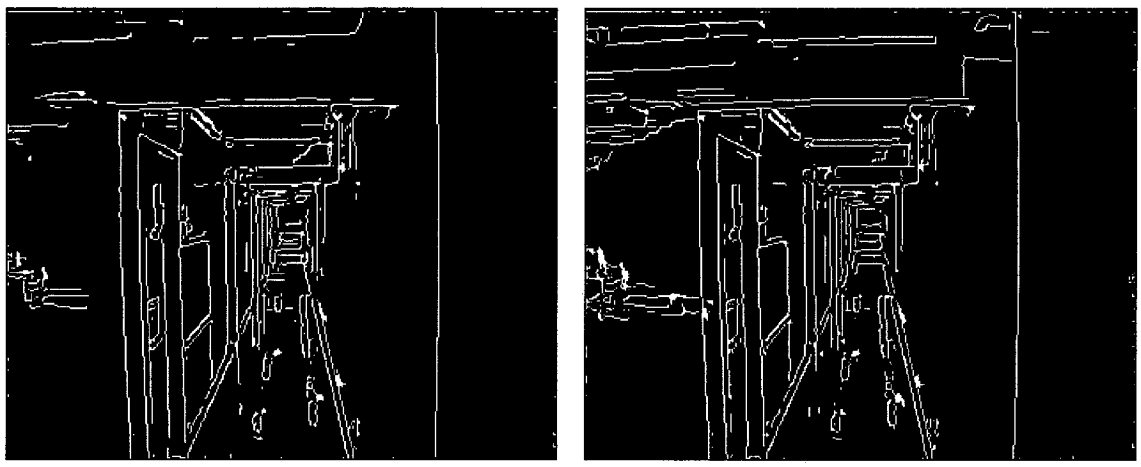

Figure 4.7. Example images from Figure 3.11 classified correctly due to edge detection.

Unfortunately, along with much of the extraneous information discarded with the edge detection, some of the details of each person is also discarded. Thus, edge images containing people with similar sizes and aspect ratios were deemed similar. Discarding all information between edges, removed some key identifiable features of the people in the images, such as skin tone. In chapter 5, we present an alternative to edge images. We still look to discard extraneous information; however, instead, we crop out only the regions of interest and discard the rest of the image. In video surveillance applications, the majority of the time the regions of interest are the portions of the image containing people. Thus, we apply a person detector on the images before inputting them to the anomaly detector.

After an edge detector is applied on an intensity image, the edge images remain as large as the original intensity images, which is approximately 102 KB. However, because so much of the information is lost in the edge detection process, and redundancies in edges are much higher, the edge detector achieved much higher compression ratios with the edge images. In fact, the edge images could be compressed to a range of 2.8-5.4 KB, with a mode and mean of 4 KB. In other words, the compressed edge images are roughly 80% smaller than the compressed intensity images. This improvement in compression ratio proves to be a significant reduction in the storage load. Although the rate of novelty decreases with time, even if we were to assume that it did not decrease, but rather stayed constant, then we would expect roughly 420 novel images every 5 days. If that were the case, at 4 KB per stored image, before filling up a 20 GB hard disk, which is a standard hard disk size, the algorithm could potentially run for over 170 years!

In order to test the speed of the algorithm with edge images, as with the intensity images, the anomaly detector was run again for 24 hours, where the only measure saved was the time to process each image. This time, 154,769 images were processed, which means that on average, each frame took approximately 0.558 seconds to process. Although, on average, the edge images took more time to process, and in 24 hours, fewer edge images were processed than intensity images, this is not an accurate gauge of computational speed. Firstly, with the edge images, more than double the novel images were found, and since the processing time of a novel image is longer and increases with the number of novel images found, it is to be expected that the edge images processed fewer images. In Figure 4.8, we observe that over 400 novel images were found during the 24 hours. We note that in Figure 3.12, after 170 novel intensity images, the anomaly detection algorithm required approximately 9 seconds to process an intensity image. In this case, after 170 novel edge images required only 5.1 seconds of processing time, which is roughly 44% less time than the processing time of the intensity images. A runtime profiler, gprof, was used to determine that 95% of the total time was spent applying the edge detector on images. In Figure 4.9, the worst-case time

of nearly 12 seconds occurs near image 160,000. This time is actually 3 seconds longer than the 9 second time of the 160,000th intensity image from Figure 3.13. However, we note that if 95% of the total time is spent in edge detection, then that means less than 0.5 seconds are spent in the actual anomaly detection. In chapter 6, we suggest ways to improve the processing time of the edge detector.

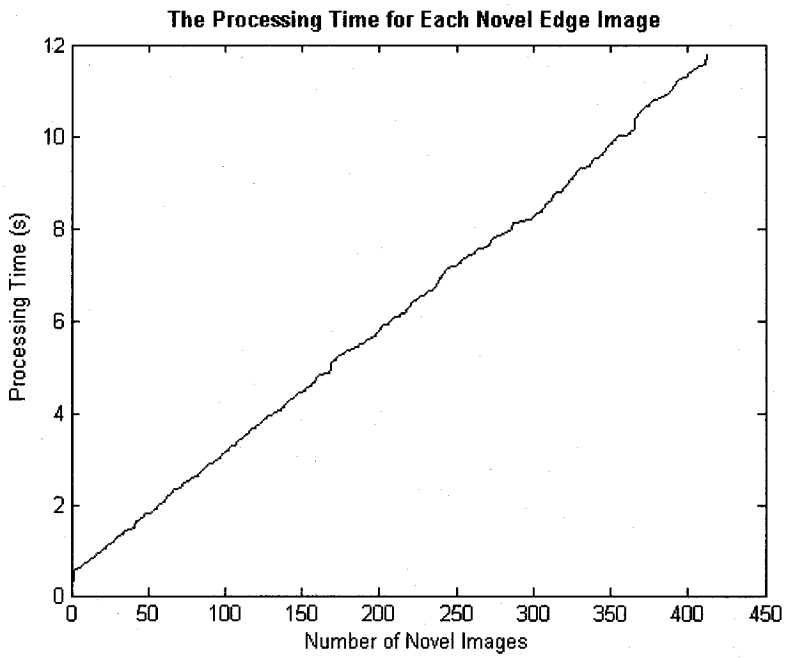

Figure 4.8. Processing time for each novel edge image.

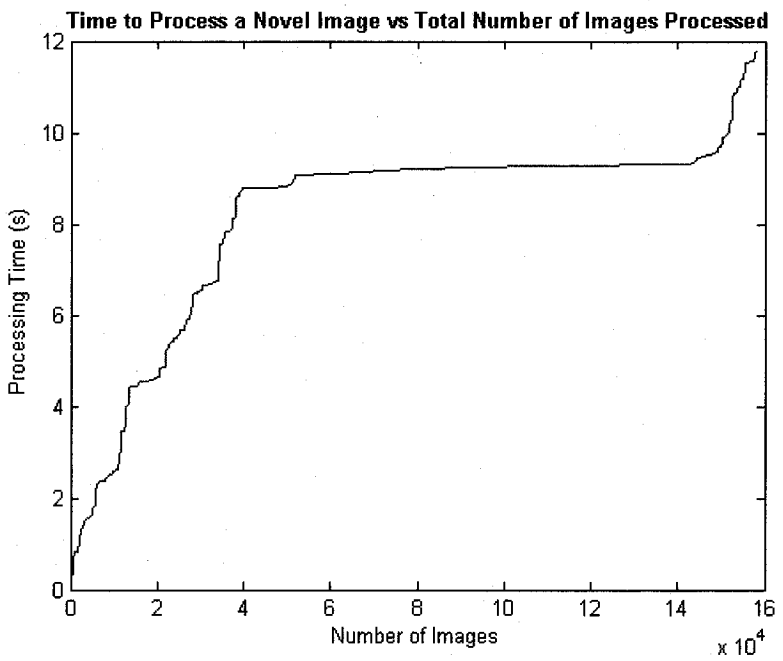

Figure 4.9. Processing time for each novel image vs. total number of images processed.

CHAPTER 5

ANOMALY DETECTION WITH PERSON DETECTION

Although the edge images already filter out a lot of information, both the intensity images and the edge images, still contain a lot of extraneous information that is not pertinent in the determination of novelty. More specifically, based on the observation that the area of interest in the image occupies less than a quarter of the total image area, we wanted to find a way of extracting only the areas of interest. Moreover, both the intensity and edge images are sensitive to slight changes in camera positions. For example, if the camera were knocked slightly out of place, then in the case where the anomaly detector is applied to either the intensity or the edge images, similar images would be deemed dissimilar, and the system would experience somewhat of a reset.

In this chapter, we present the person detector we applied to the acquired images. In section 5.2, we present the results of the tests which we ran on the demonstration system. Finally, we discuss these results in relation to the results from the tests which were run on the algorithm with intensity and edge images.

5.1 The Nair and Clark Person Detector

In our implementation, we adopt Nair and Clark's appearance-based person detector [1, 57]. In Figure 5.3, the basic structure of the person detection algorithm is shown. The input video frames are fed into the automatic labeler, which uses background subtraction to find the foreground pixels. In Figure 5.1, the resulting images from the background subtraction of the example images from Figure 1.4 are shown.

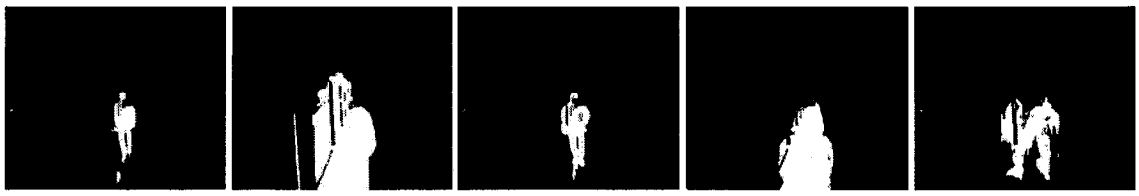

Figure 5.1. Background subtraction of example images..

The foreground pixels, which are the white pixels from the example images in Figure 5.1, are then grouped into what Nair describes as "blobs". The automatic labeler then labels the "true" label of these "blobs" as either "person" - if and only if the bounding box around it is the correct aspect ratio of a person, and at least the minimum size requirement. Otherwise, it is labeled a "non-person". After the automatic labeler labels the regions, the classifier performs the person detection by scanning a window of interest across the image.

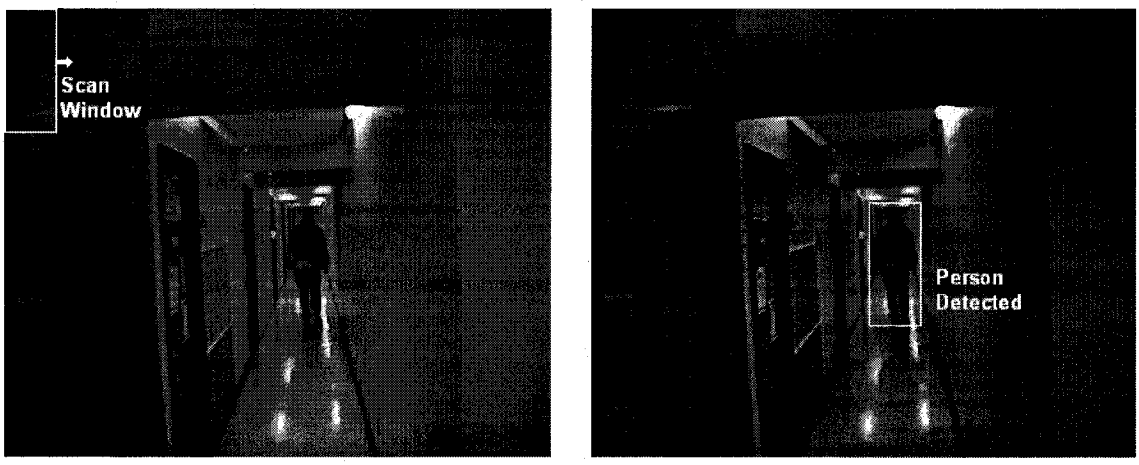

Figure 5.2. Scan window moving across image, classifier classifying each subimage as "person" or "non-person"

The algorithm computes the features within each subimage contained in the scan window, and if it matches the features already saved in the classifier, then it outputs a "predicted" label of "person". However, if the "true" label differs from the "predicted" label, then the labels and the computed classifier values are sent to the Winnow learning algorithm [49] which updates the classifier. If, after all the subimages are evaluated, the

"predicted" label was at any point labeled as "person", then the location of the person detected is output.

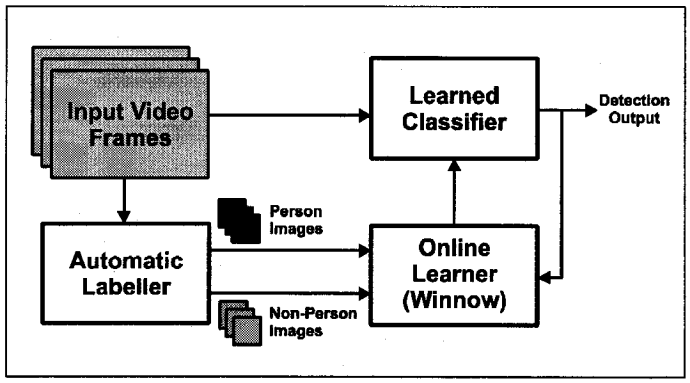

Figure 5.3. The basic structure of the Nair and Clark person detector. [1]

Once the person detector detects a person, the location of this person, specified by a bounding box around the person detected, is output. The subimage contained in the bounding box is then cropped and scaled. It is important that the person images are all scaled to an equal size in order to not skew the similarity measure. The size to which they are scaled is equally important. If they are scaled down, then some information is lost. Thus, it is more appropriate to scale all the images to the size of the largest person image. The dimensions of the images attained from the camera are 352x288 pixels. Thus, the maximum height the person image can be is 288. However, in order for the person image to exceed 200 pixels in height, the person would have to be over 7 feet tall thus, we chose the scale of 200x118 pixels. In Figure 5.4 the block diagram of the system is shown from person detection to anomaly detection.

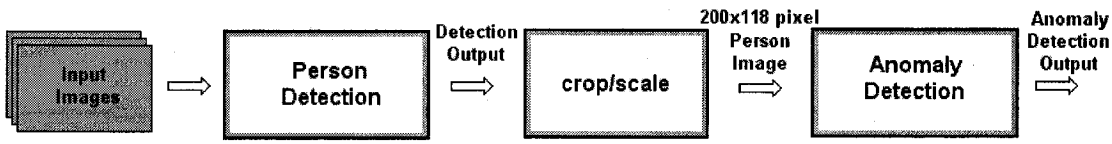

Figure 5.4. Block diagram of anomaly detector with person detection.

5.2 Results

For the purpose of clarity, we will refer to the system which applies an anomaly detector to the output of the person detector as PDAD. The PDAD was run over a 4 day period. During this time, over $3x10^5$ images were acquired from the camera and processed. In Figure 5.5, the highest similarity measure for each person image is plotted. The threshold in this case was placed at 0.09. In Figure 5.6 the number of person images versus the total number of images processed by the person detector is plotted. We observe that only 121 persons were detected. In Figure 5.7 we note that of these 121 person images, only 51 novel images were found.

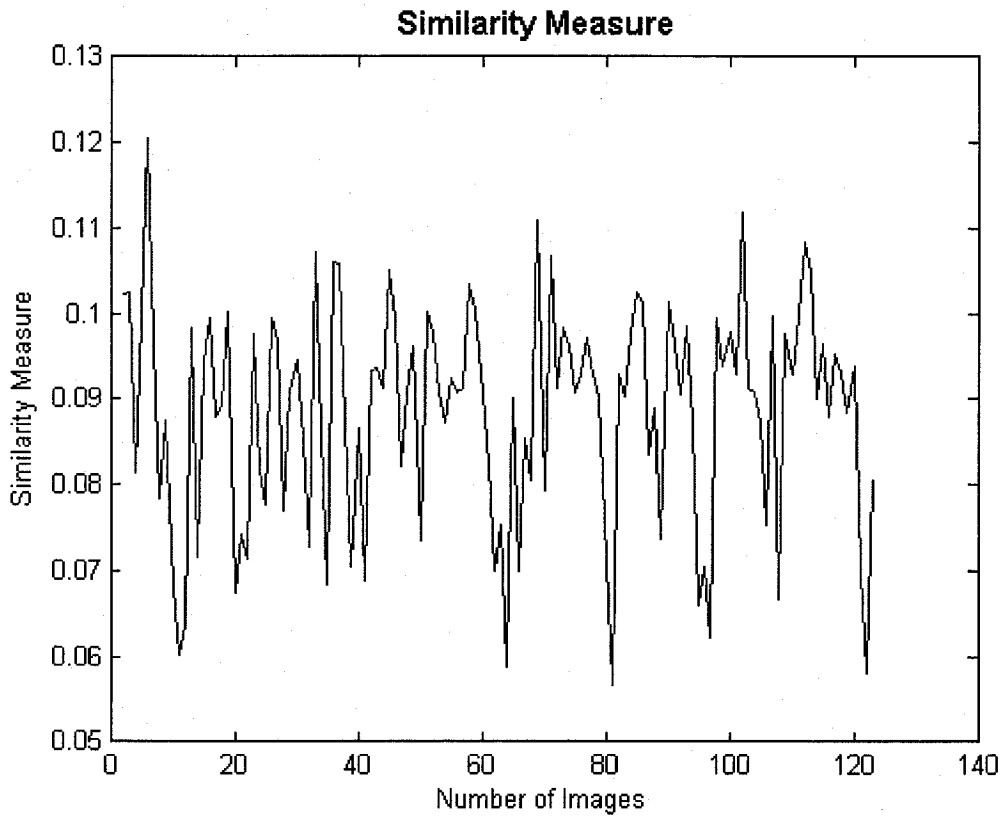

Figure 5.5. Highest similarity measure for each person image

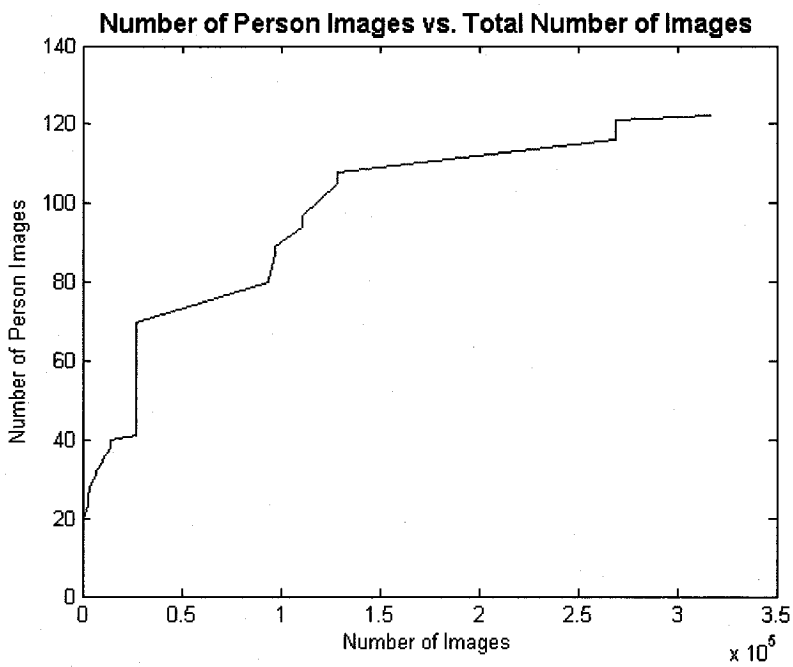

Figure 5.6. Number of Person Images vs Number of Images Processed

Figure 5.7. Number of Novel Images vs Total Number of Images

There are many advantages to applying the person detector to the images before applying the anomaly detector, namely: location invariance and reduction in storage resources. The anomaly detection algorithm is sensitive to changes in location with both the intensity and edge images. For example, in Figure 5.8 the two images, *imgA* and *imgB*, shown in a) and b) are presented again in c) and d) respectively but with the camera moved to the right. The similarity metric was applied on these images. In theory, *imgA* and *imgA_moved* should have a large amount of similarity between them. Likewise, *imgB* and *imgB_moved* should be deemed similar. However, in Table 5.1 we present the similarity measures of pairs of images. Given that the threshold for intensity images is 0.055, we note that the *imgA* and *imgA_moved*, has a similarity measure below the threshold, and thus, is considered dissimilar. Similarly, *imgB* and *imgB_moved* are considered dissimilar. What is surprising is that *imgA* is more similar to *imgB* than it is to itself moved.

	imgA_moved	***imgB_moved***	***imgA***
imgA	0.037572	0.037449	-
imgB	0.038105	0.038807	0.057810
imgA_moved	-	0.050081	-

Table 5.1. Similarity Measure for images from Figure 5.8.

By applying a person detector on the images prior to applying the anomaly detector however, a level of location invariance is achieved. In Figure 5.9, the outputs of the person detector are shown. The people from the images in Figure 5.8 were detected, cropped, and scaled to equal sizes. The similarity measure between the pairs of images is shown in Table 5.2. For the person images, a threshold of 0.09 was used, and thus we note that all the images were correctly deemed similar regardless of the location and size of the person within the image.

	imgA_moved	*imgB_moved*	*imgA*
imgA	0.169971	0.132567	-
imgB	0.134277	0.138150	0.139931
imgA_moved	-	0.133806	-

Table 5.2 Similarity measure for images from Figure 5.9.

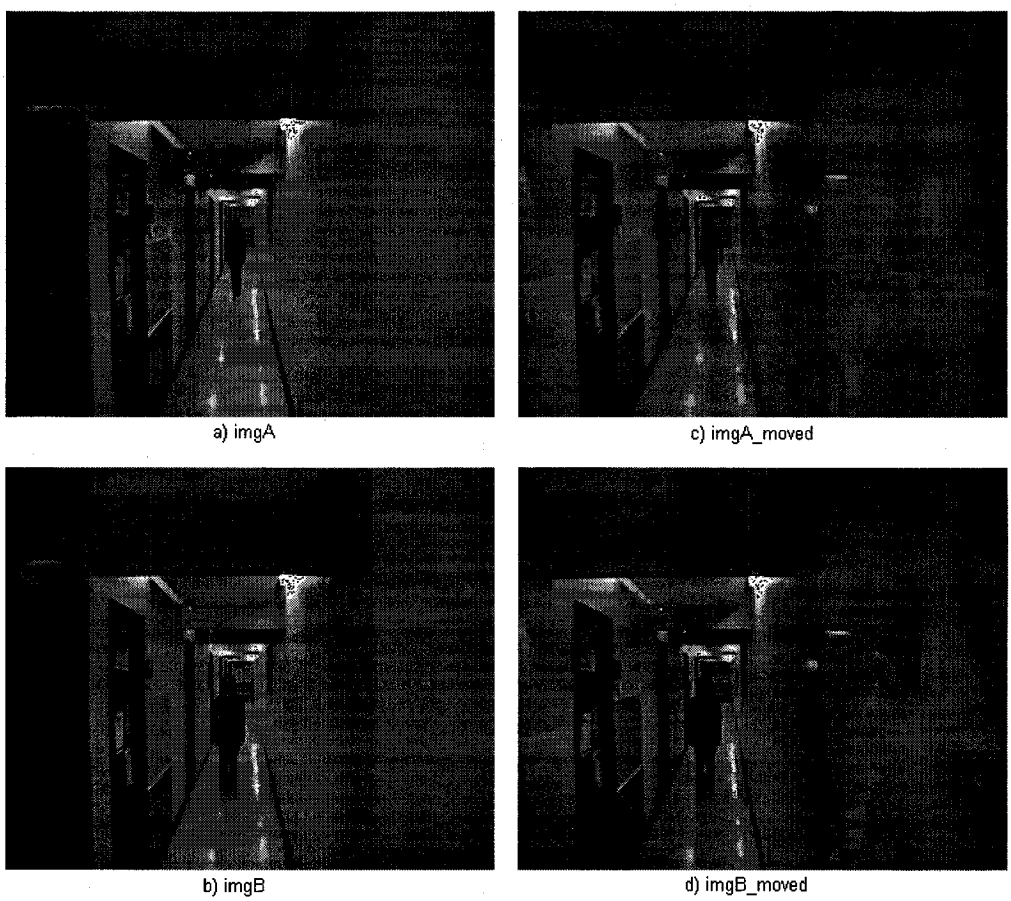

Figure 5.8. Figure from location invariance test.

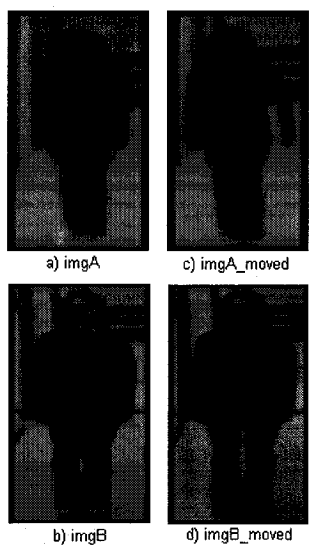

Figure 5.9. Cropped people from location invariance test.

The accuracy of the anomaly detector is dependent on the accuracy of the person detector. There were two main types of false positives with the person detector, shown in Figure 5.10. Firstly, the bounding box around the person could appear slightly off-center. The source of this error comes from the fact that the scan window of the person detector scans at fixed positions, and it returns its location once a positive result is detected. Thus, in some cases, multiple boxes can be drawn for the same person, depending on the classifier, or just one box which is slightly off center. The second type of false positives is when "non-person" instances are incorrectly classified (image on the right). Our algorithm adapts to these false positives by simply saving a copy of the "non-person" images and storing it so future false positives are not deemed novel. In chapter 6, we suggest a method that will decrease the false positive rate of the person detector.

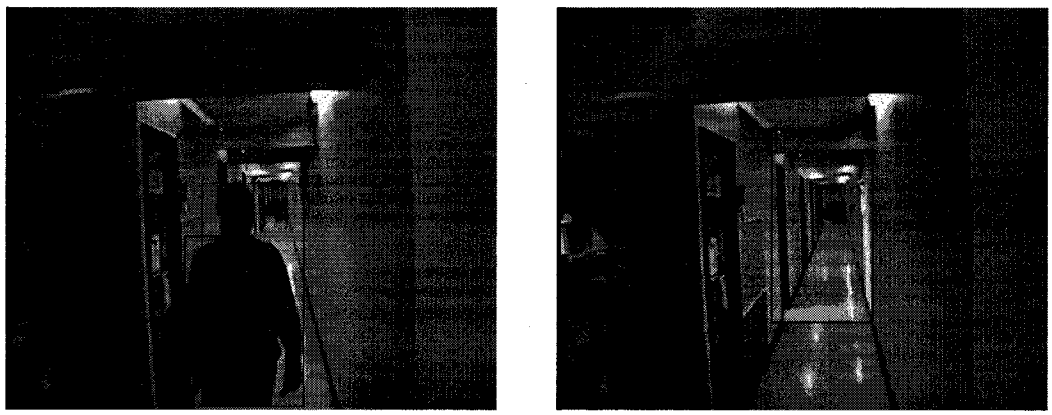

Figure 5.10. Examples of false positives for the Nair and Clark person detector.

As to be expected, applying a person detector on the images significantly reduces the size of the images that are input into the anomaly detector. As a result, the sizes of the images to be stored are smaller than the intensity images. From Figure 5.11, we see that person images are 69.5 KB, which is 30% less than the size of an intensity image. Moreover, the person image compresses to 5-13 KB, with the majority of the compressed images being around 8 KB. Thus the compressed person images are roughly 60% smaller than the compressed intensity images.

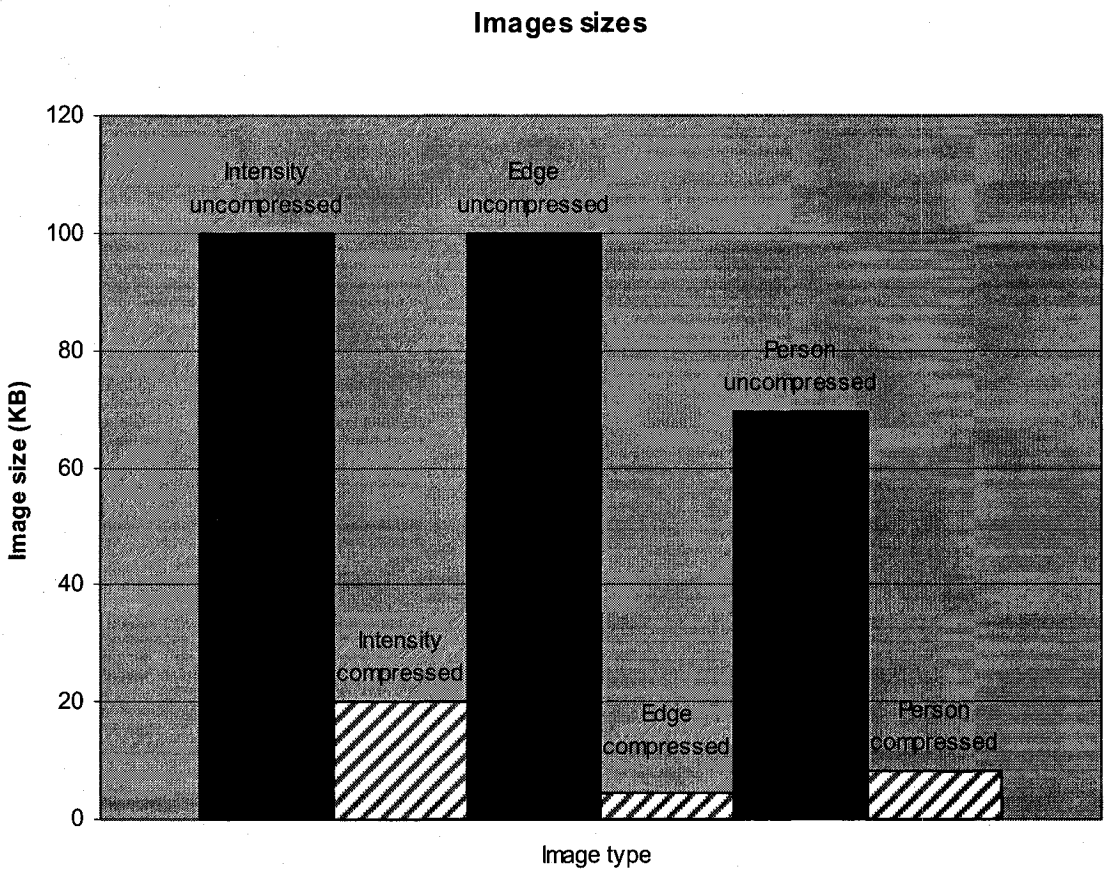

Figure 5.11 Intensity, edge and person image sizes.

In the PDAD system, the person detector algorithm is the one that calls the anomaly detector algorithm. In Figure 5.12, the processing time for each novel image detected within a 24 hour time period is plotted. The 'total time' is the total time it took to apply the person detector on the images acquired from the camera in order to make the person images, and then to apply the anomaly detector on the people images to determine novelty. The "anomaly detection time" is the time it took to apply the anomaly detector on the person images. From the plot, we see that once again, the "anomaly detection time" increased steadily with the number of novel images detected. On the other hand, while initially, the "person detection time" was erratic due to learning, it stabilized after time, and with the exception of the occasional peak, stayed at approximately 0.5 seconds/frame. These peaks are the result of the "true" label differing from the "predicted" label and therefore requiring that the classifier be updated. In Figure 5.13, we note that the time to process the person images is comparable to the time to process the intensity images. This closeness in processing times can be attributed to the fact that the portions which are cropped out are the portions containing the majority of the information of the image. Thus, in terms of computational resources, the edge images fared the best.

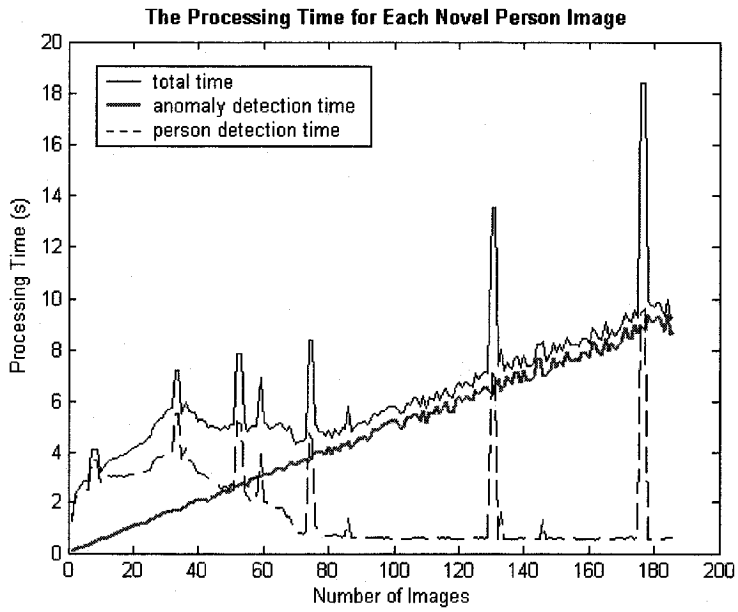

Figure 5.12. Processing time for each novel person image

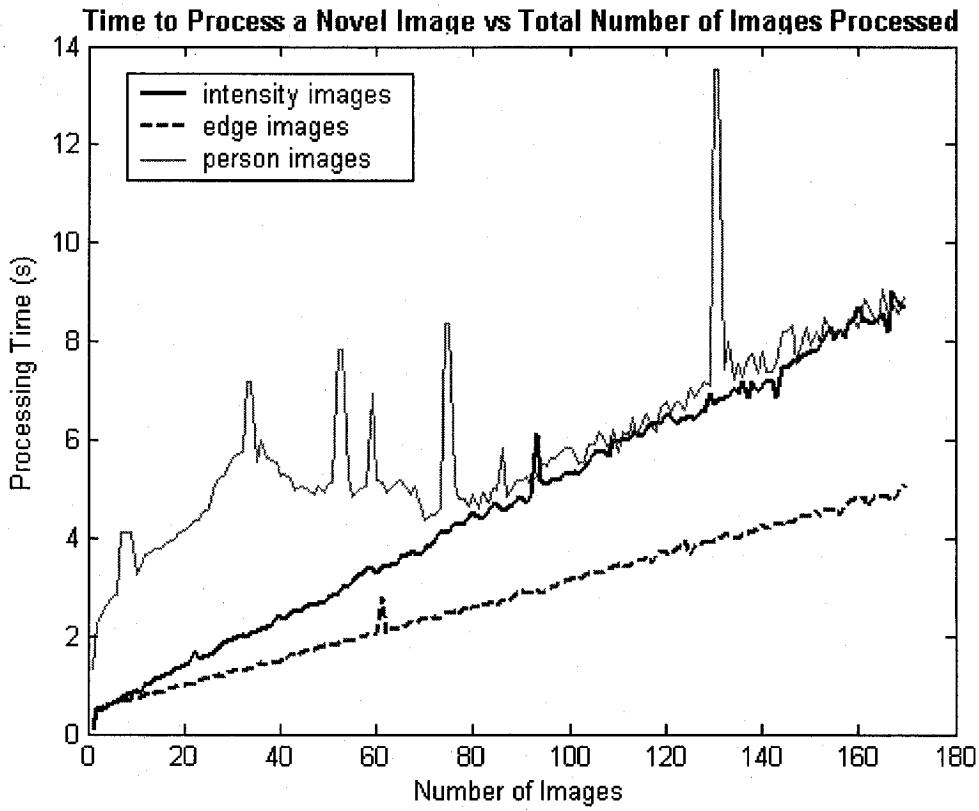

Figure 5.13. Time to process intensity, edge and person images.

In Chapter 6, further limitations of the system will be presented, as well as possible improvements to the system that could overcome the system's present shortcomings. Moreover, some future applications for this work will be summarized.

CHAPTER 6

CONCLUSION

In this thesis we have described our anomaly detection algorithm. We applied the anomaly detection on intensity, edge, and person images. In the next section, we will present some limitations of the algorithm and possible ways of improving it. Finally, we will summarize the results presented in the previous chapters and suggest some possible future applications for our compression-based anomaly detector.

6.1 Overall Limitations and Possible Improvements

In chapter 4 we discussed that it could potentially take over 170 years to fill up a 20 GB hard disk with compressed edge images. Although storage does not seem to be an issue, by reducing the number of novel images stored, we could potentially speed up the algorithm - since the time to process a novel image is directly proportional to the number of novel images. Thus, one possible improvement would be to limit the number of novel images stored. For example, a list of only the 500 most frequently matched novel images could be maintained. In this way, the processing time of a novel image has a ceiling time that it cannot exceed.

In addition to placing a cap on the worst-case processing time, it is also desirable to simply reduce the processing time of the novel-images, since even at 500 novel images, it would take 26 seconds to process an intensity image. Hardware implementation of certain tasks is one possible way of improving the processing time. For example, in chapter 4, we noticed that the task of edge detection took up over 95% of the total run time. Integrating a dedicated hardware implementation of edge detection would remove the onus of edge detection from the processor running the anomaly detector, which will free it up to focus on the task of anomaly detection. In addition, hardware implementations can be used for compressing and decompressing

the files, which could potentially also free up the processor. A camera with a higher resolution and a faster frame rate would also speed up the algorithm.

Given that the frames are processed one at a time, and the algorithm waits for one frame to be finished processing before acquiring a new image from the camera, a slow processing time for the novel images bottlenecks the process. One possible solution would be to implement a parallel process - where one process continues to acquire images from the camera, while the other process determines novelty. In this way, the gap between the times successive frames are acquired is not too wide. The parallel process could also be implemented in such a way, so that if one process was tied down checking the novelty of a novel image, the other could continue detecting for anomalies. With the person images, the algorithm could also benefit from parallel processing. Since the outcome of the person detector does not rely on the output of the anomaly detector, and the processing time of the person detector is much faster of the processing time of the anomaly detection portion, the two algorithms could run in parallel.

Another drawback of our implementation is the need to manually set the threshold level. Implementing an adaptive threshold method could greatly benefit the algorithm. The training time would be minimized, since regardless of the threshold chosen initially, the algorithm would adjust it until it was correct. Moreover, it would solve the issue of illumination sensitivity with the intensity images.

In [1], Ahmedali presented a multi-camera collaboration for person detection. In his approach, he used the information gathered from multiple cameras to reduce the false positive rates seen in Nair and Clark's system. A more accurate person detector would lead to an improvement in accuracy and speed of our algorithm, since false positives increase the number of comparisons.

6.2 Future Applications and Conclusion

We have presented a method for anomaly detection for video surveillance applications. In our approach, we used a compression-based similarity measure which compared the compression sizes of two images for the purpose of determining the level of similarity between them. Input images dissimilar from all the previously-stored images, were considered novel, and these images were then stored for comparison to future incoming images. In our attempt to make a reliable real-world application, we focused on the issues of storage and computational resources, in addition to accuracy, low-cost, adaptability, and scalability. The compression-based method inherently led to reductions in storage loads. In order to improve the speed and sensitivity of the algorithm, we applied the anomaly detector to two image features, edges and people. An edge detector was applied on all input images. The edge detector was faster and required less storage space than the intensity images. However, it also discarded much information about the regions of interest, namely, the persons in the images. Thus, we looked at the effects of applying a person detector to the images, which focused only on these regions of interest. By homing in on only these regions of interest, the person detector effectively increased the level of accuracy by removing a lot of the algorithm's sensitivity to location. However, because regions of interest contained most of the image's pertinent information, and because the person detection process was an extra step required, then the overall processing time of a novel image was not improved from the processing time of a novel intensity image. Thus, in the previous section, we suggested ways in which the speed of this algorithm could be improved.

As shown in this thesis, the compression-based anomaly detection algorithm was an effective method of determining novelty in three types of images: intensity, edges, and persons. This approach can also be extended to face images. While with person images, much of the extraneous information is already discarded, they still contain varying information which can skew the results of the anomaly detector. Clothing changes from day to day and season to season. In fact, winter apparel can

even change the aspect ratio of the person, as people tend to wear thicker clothes and extra layers. Moreover, people are often not walking empty-handed, and the objects which they are holding can also lead to variations which can skew results. The head has four degrees of freedom within the neck. Thus, the face, in relation to the body also affects the result. Faces also have numerous variations, such as beards, glasses, and facial expressions; however, by eliminating the rest of the body from the image, much of the variance is eliminated. In addition, with multi-camera collaborations, the viewing angle of the camera is less important, as the face can be captured from multiple angles, and the image with the best face image can be output to the anomaly detector. In other words, no matter which way the head is turned, with a multi-camera system, the best viewing angle for the face image can be captured. Thus, Ahmedali's multi-camera person detector [1] could be adapted to output the portions of the images containing faces rather than people.

One interesting potential application for our anomaly detection algorithm, is re-run detection for digital video recorders (DVR), such as TiVo [39]. A DVR is a device which allows a user to record television programs to its internal hard disk for future viewing. By storing the first few frames of all the shows viewed by the user, the DVR could use our anomaly detector to compare the frames of programs scheduled for recording, and only record the shows the user has not already watched.

References

[1] T. Ahmedali. "Collaborative multi-camera surveillance using automated person detection." Thesis, McGill University,

[2] C. Arcelli and G. S. Di Baja. "Thinning Algorithm Based on Prominence Detection". *Pattern Recognition*, vol. 13, pp. 225-235, 1981.

[3] P. Aswani, K. K. Wong, and M. N. Chong, "Tracking of deformable objects." in Proceedings of SPIE - The International Society for Optical Engineering, 2001, pp. 476-485.

[4] T. Bell and D. Kulp. "Longest-match string searching for Ziv-Lempel Compression". *Software - Practice and Experience*, vol. 23, pp. 757-772, 1993.

[5] C. H. Bennett, M. Li, and B. Ma. "Chain Letters and Evolutionary Histories". *Scientific American*, vol. 288, pp. 79-81, 2003.

[6] M. Bogaert, N. Chleq, P. Cornez, C. S. Regazzoni, A. Teschioni, and M. Thonnat, "The PASSWORDS project." in IEEE International Conference on Image Processing, 1996, pp. 675-678.

[7] N. Bouaynaya and D. Schonfeld, "A complete system for head tracking using motion-based particle filter and randomly perturbed active contour." in Proceedings of SPIE - The International Society for Optical Engineering, 2005, pp. 864-873.

[8] S. Bunton, "Structure of DMC." in Proceedings of Data Compression Conference, 1995, pp. 72-81.

[9] S. Bunton, "Generalization and improvement to PPM's 'blending'." in Proceedings from Data Compression Conference, 1997, pp. 426.

[10] S. Bunton, "Bayesian state combining for context models." in Proceedings of the Data Compression Conference, 1998, pp. 329-338.

[11] M. Burrows and D. J. Wheeler. "A block-sorting Lossless Data Compression Algorithm". *SRC Research Report*, 1994.

[12] J. Canny. "Computational Approach to Edge Detection". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[13] R. T. Chin, H.-K. Wan, D. L. Stover, and R. D. Iverson. "One-pass Thinning Algorithm and its Parallel Implementation". *Computer Vision, Graphics, and Image Processing*, vol. 40, pp. 30-40, 1987.

[14] N. Chleq and M. Thonnat, "Realtime image sequence interpretation for video-surveillance applications." in IEEE International Conference on Image Processing, 1996, pp. 801-804.

[15] Christias, P., "UNIX man pages: cat," [Online document], Available at: http://unixhelp.ed.ac.uk/CGI/man-cgi?cat, 1994

[16] J. Cleary and I. Witten. "Data Compression Using Adaptive Coding and Partial String Matching". *IEEE Transactions on Communications*, vol. 32, pp. 396-402, 1984.

[17] D. Corall, "VIEWS: Computer Vision for surveillance applications," in *IEEE Colloquium on Active and Passive Techniques for 3D Vision*, vol. 8. London, 1991.

[18] G. V. Cormack and R. N. S. Horspool. "Data compression using dynamic Markov modelling". *The Computer Journal*, vol. 30, pp. 541-550, 1987.

[19] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz. "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks". *Expert Systems with Applications*, vol. 29, pp. 713-722, 2005.

[20] A. Deruyver, Y. Hode, and L. Soufflet, "Segmentation technique for cerebral NMR images." in IEEE International Conference on Image Processing, 1994, pp. 716-720.

[21] C. P. Diehl. "Toward Efficient Collaborative Classification for Distributed Video Surveillance." Ph.D. Thesis, Carnegie Mellon University,

[22] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction." in Proceedings of 6th European Conference on Computer Vision (ECCV 2000), 2000, pp.

[23] E. R. Fiala and D. H. Greene. "Data compression with finite windows". *Communications of the ACM*, vol. 32, pp. 490-505, 1989.

[24] B. E. Flinchbaugh and T. J. Olson, "Autonomous video surveillance." in Proceedings of SPIE - The International Society for Optical Engineering, 1997, pp. 144-151.

[25] D. A. Forsyth and M. M. Fleck. "Automatic detection of human nudes". *International Journal of Computer Vision*, 1999.

[26] T. Frank, M. Haag, H. Kollnig, and H. H. Nagel, "Tracking of occluded vehicles in traffic scenes." in Proceedings of 4th European Conference on Computer Vision (ECCV 1996), 1996, pp. 485-494.

[27] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of 13th International Conference in Machine Learning*. Bari, Italy, Jul. 3-6 1996.

[28] D. B. Gennery. "Visual tracking of known three-dimensional objects". *International Journal of Computer Vision*, vol. 7, pp. 243-270, 1992.

[29] J. Gilchrist, "Parallel data compression with bzip2." in Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems, 2004, pp.

[30] G. Giraudon, "Edge Detection from Negative Maximum of Second Derivative." in Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 1985, pp. 643-645.

[31] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site." in, 1998, pp. 22-29.

[32] P. C. Gutmann and T. C. Bell, "Hybrid approach to text compresion." in Proceedings of IEEE Data Compression Conference, 1994, pp. 225-233.

[33] I. Haritaoglu, D. Harwood, and L. S. Davis, "W^4: Who? When? Where? What? A real time system for detecting and tracking people." in Proceedings of 3rd IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 222-227.

[34] D. T. Hoang, P. M. Long, and J. S. Vitter. "Dictionary selection using partial matching". *Information Sciences*, vol. 119, pp. 57-72, 1999.

[35] R. N. Horspool and G. V. Cormack, "Dynamic Markov modelling - A prediction technique." in Proceedings of the Hawaii International Conference on System Science, 1986, pp. 700-707.

[36] P. G. Howard and J. S. Vitter. "Arithmetic coding for data compression". *Proceedings of the IEEE*, vol. 82, pp. 857-865, 1994.

[37] D. A. Huffman, "A Method for Construction of Minimum-Redundancy codes." in Proceedings of the Institute of Radio Engineers, September, 1952, pp. 1095-1101.

[38] A. Iannino and S. D. Shapiro, "Iterative Generalization of the Sobel Edge Detection Operator." in Proceedings - IEEE Computer Society Conference on Pattern Recognition and Image Processing, 1979, pp. 130-137.

[39] TiVo Inc., "The TiVo Homepage," [Online document], Available at: http://www.tivo.com/0.0.asp, 1998-2006 [February 18 2006].

[40] Y. A. Ivanov and A. F. Bobick, "Recognition of multi-agent interaction in video surveillance." in Proceedings of the IEEE International Conference on Computer Vision, 1999, pp. 169-176.

[41] S. Jahangir, N. Tanveer, M. Ahmad, and S. A. Khan, "Networked surveillance system using object-centric motion-based tracking and classification." in Student Conference on Engineering Sciences and Technology, SCONEST 2004, 2004, pp. 106-111.

[42] T. Kanade, R. T. Collins, A. J. Lipton, P. Burt, and L. Wixson, "Advances in cooperative multi-sensor video surveillance." in Proceedings, DARPA Image Understanding Workshop, 1998, pp. 3-24.

[43] N. Kanopoulos, N. Vasanthavada, and R. L. Baker. "Design of an Image Edge Detection Filture Using the Sobel Operator". *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 358-367, 1988.

[44] D. Koller, K. Daniilidis, and H. H. Nagel. "Model-based object tracking in monocular image sequences of road traffic scenes". *International Journal of Computer Vision*, 1993 257-281.

[45] C. Kruegel and G. Vigna, "Anomaly detection of Web-based attacks." in Proceedings of the ACM Conference on Computer and Communications Security, 2003, pp. 251-261.

[46] R. Laganiere and R. Elias, "The detection of junction features in images." in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2004, pp. 573-576.

[47] G. G. Langdon. "An introduction to arithmetic coding". *IBM Journal of Research and Development*, vol. 28, pp. 135-149, March 1984.

[48] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitanyi. "The similarity metric". *IEEE Transactions on Information Theory*, 2004.

[49] N. Littlestone. "Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm". *Machine Learning*, vol. 2, pp. 285-318, 1988.

[50] D. G. Lowe. "Robust model-based motion tracking through the integration of search and estimation". *International Journal of Computer Vision*, vol. 8, pp. 113-122, 1992.

[51] Elysium Ltd. and 2KAN, "JPEG Homepage," [Online document], Available at: http://www.jpeg.org/jpeg/index.html, 2004 [February 2nd 2006].

[52] M. Mandjes, I. Saniee, and A. L. Stolyar. "Load characterization and anomaly detection for voice over IP traffic". *IEEE Transactions on Neural Networks*, vol. 16, pp. 1019-1026, 2005.

[53] D. Marr and E. C. Hildreth. "Theory of Edge Detection". *Proceedings of the Royal Society of London*, vol. B207, pp. 187-217, 1980.

[54] R. Mehrotra and S. Nichani. "Corner detection". *Pattern Recognition Letters*, vol. 23, pp. 1223-1233, 1990.

[55] F. G. Meyer and P. Bouthemy. "Region-based tracking using affine motion models in long image sequences". *Computer Vision Graphics, Image Proc.: Image Understanding*, vol. 60, pp. 119-140, 1994.

[56] V. S. Miller and M. N. Wegman. "Variations on a theme by Ziv and Lempel". *Combinatorial Algorithms on Words*, vol. 12, pp. 131-140, 1985.

[57] V. Nair and J. J. Clark, "An unsupervised, online learning framework for moving object detection." in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. 317-324.

[58] V. Nair and J. J. Clark, "Automated Visual Surveillance Using Hidden Markov Models." in Proceedgins of the 15th Vision Interface Conference, May 2002, pp.

[59] J. A. Noble. "Finding Corners". *Image and Vision Computing*, vol. 6, pp. 121-128, 1988.

[60] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates." in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 1997), 1997, pp 192-199., pp.

[61] C. Papageorgiou and T. Poggio, "Pattern classification approach to dynamical object detection." in Proceedings of the IEEE International Conference on Computer Vision, 1999, pp.

[62] J. M. S. Prewitt. "Object Enhancement and Extraction". *In Picture Processing and Psychopictorics*, pp. 75-149, 1970.

[63] J. Rissanen. "Generalized Kraft inequality and arithmetic coding." *IBM Journal of Research and Development*, vol. 20, pp. 198-203, 1976.

[64] J. Rissanen and G. G. Langdon. "Arithmetic coding". *IBM Journal of Research and Development*, vol. 23, pp. 149-162, 1979.

[65] K. Rohr. "Recognizing corners by fitting parametric models". *International Journal of Computer Vision*, vol. 9, pp. 213-230, 1992.

[66] A. Rosenfeld and M. Thurston. "Edge and Curve Detectio nfor visual Scene Analysis." *IEEE Transactions on Computer*, vol. 20, pp. 562-569, 1971.

[67] E. Salari and P. Siy. "Ridge-seeing Method for Obtaining the Skeleton of Digital Images". *IEEE Transactions Systems, man and cybernetics*, vol. 14, pp. 524-528, 1984.

[68] K. S. Shanmugam, F. M. Dickey, and J. A. Green. "Optimal Frequency Domain Filter for Edge Detection in Digital Pictures". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, pp. 37-49, 1979.

[69] D. Shkarin, "PPM: one step to practicality." in Proceedings of the IEEE Data Compression Conference, 2002, pp. 202-211.

[70] R. W. Smith. "Computer Processing of Line Images: A Survey". *Pattern Recognition*, vol. 20, pp. 7-15, 1987.

[71] A. S. Sodiya, H. O. D. Longe, and A. T. Akinwale. "Maintaining privacy in anomaly-based intrusion detection systems". *Information Management and Computer Security*, vol. 13, pp. 72-80, 2005.

[72] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking." in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999, pp. 246-252.

[73] J. A. Storer and T. G. Szymanski. "Data compression via textual substitution". *Journal of the ACM*, vol. 29, pp. 928-951, 1982.

[74] J. Teuhola and T. Raita. "Application of a finite-state model to text compression". *The Computer Journal*, vol. 36, pp. 607-614, 1993.

[75] M. Thottan and J. Chuanyi. "Anomaly detection in IP networks". *IEEE Transactions on Signal Processing*, vol. 51, pp. 2191-2204, 2003.

[76] P. Viola and M. J. Jones. "Robust Real-Time Face Detection". *International Journal of Computer Vision*, 2004.

[77] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *Proceedings of the IEEE International Conference on Computer Vision*. Nice, France, 2003.

[78] P. A. J. Volf and F. M. J. Willems, "Switching between two universal source coding algorithms." in Proceedings of the IEEE Data Compression Conference, 1998, pp. 491-500.

[79] L. Wang, M. F. Ramoni, K. D. Mandl, and P. Sebastiani. "Factors affecting automated syndromic surveillance". *Artificial Intelligence in Medicine*, vol. 34, pp. 269-278, 2005.

[80] F. Wei, M. Miller, S. J. Stolfo, L. Wenke, and P. K. Chan, "Using artificial anomalies to detect unknown and known network intrusions." in Proceedings of IEEE International Conference on Data Mining, 2001, pp. 123-130.

[81] T. A. Welch. "A technique for high-performance data compression". *IEEE Computer*, vol. 17, pp. 8-18, 1984.

[82] R. N. Williams, "An extremely fast Ziv-Lempel data compression algorithm." in Proceedings of the IEEE Data Compression Conference, 1991, pp. 362-371.

[83] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, "Rule-based anomaly pattern detection for detecting disease outbreaks." in Proceedings of the National Conference on Artificial Intelligence, 2002, pp. 217-223.

[84] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, "Bayesian Network Anomaly Pattern Detection for Disease Outbreaks." in Proceedings on Twentieth International Conference on Machine Learning, 2003, pp. 808-815.

[85] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner. "What's Strange About Recent Events (WSARE): An algorithm for the early detection of disease outbreaks". *Journal of Machine Learning Research*, vol. 6, pp. 1961-1998, 2005.

[86] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. "Pfinder: real-time tracking of the human body". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997.

[87] Y. Wu, T. S. Huang, and K. Toyama, "Self-supervised learning for object recognition based on kernel discriminant-EM algorithm." in Proceedings of the IEEE International Conference on Computer Vision, 2001, pp. 275-280.

[88] Y. Yang and F. Ma, "An Unsupervised Anomaly Detection Patterns Learning Algorithm." in International Conference on Communication Technology Proceedings, ICCT, 2003, pp. 400-402.

[89] J. Ziv and A. Lempel. "Universal algorithm for sequential data compression". *IEEE Transactions on Information Theory*, vol. IT-23, pp. 337-343, 1977.

[90] J. Ziv and A. Lempel. "Compression of individual sequences via variable-rate coding". *IEEE Transactions on Information Theory*, vol. IT-24, pp. 530, 1978.