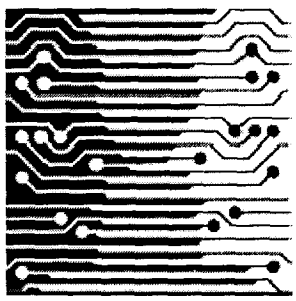
Anomaly Detection and Prediction of Human Actions in a Video Surveillance Environment

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF SCIENCE
AT THE UNIVERSITY OF CAPE TOWN
IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Nemanja Spasic
November 2007

Supervised by
Dr. Anet Potgieter



Acknowledgements

First and foremost I thank my parents for pushing me on through this dissertation and for all their much needed support. Many thanks go to my grandmother for her words of wisdom and encouragement, my girlfriend for her patience whilst listening to all my crazy ideas, my supervisor for her advice and never-ending ability to grasp all the concepts that I threw at her, past and present employees of Complex Adaptive Systems (CAS) for their constructive criticism and ideas and finally to the members of the Agents Research Lab.

Abstract

World wide focus has over the years been shifting towards security issues, not in least due to recent world wide terrorist activities. Several researchers have proposed state of the art surveillance systems to help with some of the security issues with varying success. Recent studies have suggested that the ability of these surveillance systems to learn common environmental behaviour patterns as wells as to detect and predict unusual, or anomalous, activities based on those learnt patterns are possible improvements to those systems. In addition, some of these surveillance systems are still run by human operators, who are prone to mistakes and may need some help from the surveillance systems themselves in detection of anomalous activities.

This dissertation attempts to address these suggestions by combining the fields of Image Understanding and Artificial Intelligence, specifically Bayesian Networks, to develop a prototype video surveillance system that can learn common environmental behaviour patterns, thus being able to detect and predict anomalous activity in the environment based on those learnt patterns. In addition, this dissertation aims to show how the prototype system can adapt to these anomalous behaviours and integrate them into its common patterns over a prolonged occurrence period.

The prototype video surveillance system showed good performance and ability to detect, predict and integrate anomalous activity in the evaluation tests that were performed using a volunteer in an experimental indoor environment. In addition, the prototype system performed quite well on the PETS 2002 dataset 1, which it was not designed for. The evaluation procedure used some of the evaluation metrics commonly used on the PETS datasets. Hence, the prototype system provides a good approach to anomaly detection and prediction using Bayesian Networks trained on common environmental activities.

Keywords: Image Understanding, Artificial Intelligence, Bayesian Networks, Video Surveillance, Anomaly Detection and Prediction.

Contents Page

List of Figures

Page

List of Tables

Page

List of Acronyms

AI	Artificial Intelligence
BM	Background Model
BN	Bayesian Network
CAS	Complex Adaptive System
CPT	Conditional Probability Table
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
DOG	Dynamic Oriented Graph
EM	Expectation Maximisation
FN	False Negative
FP	False Positive
FPE	Foreground Pixel Extraction
FPS	Frames per Second
GHz	GigaHertz
HHM	Hidden Markov Model
IU	Image Understanding
JPD	Joint Probability Distribution
MCMC	Markov Chain Monte Carlo
MDL	Minimum Description Length
MLE	Maximum Likelihood Estimate
MJPEG	Motion JPEG
NN	Neural Network
PDF	Probability Density Function
PETS	Performance Evaluation of Tracking and Surveillance, annual IEEE workshop

RAM	Random Access Memory
TD	Temporal Differencing
TN	True Negative
TP	True Positive
UCT	University Of Cape Town
XMLBIF	Mark-up Language (XML) based Bayesian Network Interchange Format

Chapter 1

Introduction

Just a few years ago surveillance systems used to be state of the art technology, limited to top secret military operations. However, nowadays surveillance systems are becoming the norm in modern day society. Terror and fear have caused most busy environments to be monitored in one way or another. A mixture of security personnel and cameras are scattered across many city centres, shopping malls, mass transportation building, sports events and even schools.

Massive amounts of research and money are invested into surveillance systems by countries and governments in order to detect unwanted or unexpected behaviour. Such behaviour can be referred to as *anomalous* behaviour. The latest of these surveillance systems to be implemented in Cape Town is the automatic vehicle license plate recognition system that will aid the police force in identifying people who break the rules of the road.

However, surveillance systems are not limited to security applications alone. Several sporting disciplines, e.g. golf, have used surveillance systems in an attempt to improve the ability of each player. Other sports disciplines, like cricket, have used surveillance systems to try and predict the opponent's next move. Surveillance systems have also been used to monitor and aid elderly people in nursing homes.

Although a number of surveillance systems use some form of prediction in their implementation, the majority of surveillance systems still passively monitor their environments and only raise alarms when anomalous behaviour is recognized or detected. Such sudden alarms do not give the response team enough time to prepare themselves and by the time they arrive at the scene it may be too late to stop the perpetrator(s). Other surveillance systems do not even raise alarms, but instead they only keep recordings of events which then have to be analysed by human operators. This is a very laborious and error prone task even for the operators that are fully focused on the task at hand.

These realities clearly indicate that there is a need in modern society for surveillance systems that can aid surveillance systems' operators by predicting anomalous behaviours before they happen, so that the necessary steps can be taken in time to stop or prevent such behaviours. The previous statement forms the basis of this research.

1.1 Motivation

In recent years there has been an increased interest in computer vision and in particular video surveillance systems. In a recent overview of existing video surveillance systems and techniques, Hu *et al.* [5] stressed the point that although there has been a vast amount of research done in video surveillance "many issues are still open and deserve further research". In addition, Hu *et al.* [5] state that there is scope in research and a general need for surveillance systems that can understand behaviours in an environment, as well as ones that can detect and predict unusual, or *anomalous,* behaviour in those environments. Hu *et al.* [5] also state that such anomalous behaviour should not only be specified according to the functionality of the surveillance system, e.g. a traffic monitoring system detecting traffic accidents or car theft, but should also predict what will happen in the near future based on the current state of the environment.

Stauffer and Grimson [4] highlight this fact by stating that "anomaly detection and classification could be greatly enhanced by learning context cycles". As an example they state that if their traffic monitoring system could learn traffic light cycles then cars that pass through red traffic lights could be detected as unusual. In addition, by learning day and night delivery patterns of transport vehicles separately they could also detect that night time deliveries are unusual.

At the time of writing this dissertation, little research had been devoted to anomaly prediction of human behaviour by video surveillance systems, particularly with the use of Artificial Intelligence techniques such as Bayesian Networks as a modelling and prediction tool. Hence, there is scope for research about the prediction of anomalous human behaviour using Bayesian Networks in video surveillance systems.

1.2 Research Questions

From the previous discussion about the motivation for research the following research questions were formulated:

1. Can a Bayesian Network be used to model the environment and peoples' behaviour in that environment?
2. How can the massive amount of data needed to model an environment be represented so that a Bayesian Network can be used to model the environment without sacrificing vital information about the environment?
3. Can anomalous behaviour be detected from learnt behavioural patterns?
4. Can predictions of human behaviour be done in near real time?

1.3 Contributions of the Research

The main contribution of this research is in the field of Artificial Intelligence, specifically:

1. Combining the use of a collection of Bayesian Networks and Image Understanding in a video surveillance system, in order to model an environment and the behaviours that are happening in that environment;
2. Detecting and predicting human behaviour based on the learnt patterns, some of which could be anomalous activity;
3. Finally, and most importantly, the research contributes to the body of knowledge in surveillance systems by providing support for the use of Bayesian Networks in surveillance systems as well as to the body of knowledge in data pattern learning using Bayesian Networks.

In addition, the research conducted during this dissertation has several application areas, some of which include:

- Security Areas:
    - Mass transport buildings e.g. Airports, Railways, Bus Terminus
    - Shopping malls
    - Crowded sports events
    - Naval bases
- Helping Areas
    - Old age homes
    - Sport disciplines

- Monitoring Areas
    - Schools
    - Libraries
    - Traffic systems
    - City Centres

1.4 Scope and Limitations

The aim of this research is to demonstrate the detection and prediction of anomalous behaviour in an environment through the use of a prototype Bayesian Network based video surveillance system. The prototype system that was developed can only handle a subset of possible events in an environment and by no means is it a fully fledged surveillance system, but instead it is a tool that is used as a concept demonstrator.

In particular, the surveillance system can only monitor individuals in the environment and is limited to the following actions: *standing, walking and running*. Furthermore, the predictions that were made during the evaluation of the prototype were dependent on third-party software, namely BaBe [62], which could have influenced the results that were obtained. However, upon analysing the results this does not appear to be the case.

Finally, the research conducted on the use of a collection of Bayesian Networks to model human actions during this dissertation is assumed to be unique work. Although no similar research was found, this fact cannot be completely guaranteed. If similar research does exist then the methods used could be compared and assessed as a topic for future work.

1.5 Prototype Implementation

To demonstrate the research that was conducted, a prototype video surveillance system was implemented to monitor human behaviour (walking, standing and running) in a specific environment, namely the Agents Research Lab of The University of Cape Town (UCT). The environment was divided into a grid of blocks so that positional information could be used to determine the location of each behaviour that happened in the environment. The prototype comprised of two major components, namely the Image Understanding (IU) component and the Artificial Intelligence (AI) component.

The IU component was designed as a data capturing component and was responsible for extracting data from images that were captured by a video camera. This data was then processed by the AI component in order to monitor the people in the environment and their behaviours. At the core of the AI component was a Bayesian Network (BN) which detected anomalous behaviour based on learnt behaviour patterns. In addition, the users of the prototype surveillance system could add their own anomalous actions to override the learnt patterns as well as specify the restricted areas as any of the grid blocks.

1.6 Dissertation Outline

The rest of the dissertation is structured as follows:

- **Chapter 2: Background**
  Related background topics are presented in the fields of Image Understanding and Artificial Intelligence that are specific to developing a video surveillance system. The background on Bayesian Networks is of particular significance.
- **Chapter 3: Related Research**
  Related research in surveillance systems is presented as well as the application of Bayesian Networks in those surveillance systems.
- **Chapter 4: Research Methodology**
  The methodology that was used during this research is presented as well as the implementation of the prototype surveillance system.
- **Chapter 5: Results and Evaluation**
  The evaluation tests and results that were obtained are presented in this Chapter. In addition, an evaluation of the system with the PETS 2002 [41] dataset 1 is presented.
- **Chapter 6: Conclusion and Future Work**
  The conclusion to the dissertation is presented in this Chapter and possible directions of future work are discussed.

Chapter 2

Background

Most existing video surveillance systems combine the fields of Image Understanding (IU) and Artificial Intelligence (AI) in their implementations. Hence, this Chapter discusses research techniques and methods in both IU and AI that are required to understand the research carried out during this dissertation and the prototype surveillance system that was developed to demonstrate that research.

The first part of the Chapter focuses on the field of IU. The basics of IU are presented in Appendix C. More advanced IU topics are presented in this chapter with advantages, disadvantages and points to note about each topic being discussed. On some of the possible ways to evaluate each IU topic is presented in Appendix D.

The second part of the Chapter focuses on the field of AI. This discussion starts off with probability theory, followed by modelling and knowledge representation, probabilistic reasoning and Bayesian Networks (BN)s. The discussion concludes with the presentation of BaBe [62].

2.1 Image Understanding (IU)

This section presents some of the research techniques in the field of IU that are needed to understand the implementation of the video surveillance prototype during this dissertation. The IU basics are described in Appendix C.

2.1.1 Image Understanding Overview

What is image understanding?

According to Gonzalez and Woods [47], Image Understanding (IU) is the process of applying Artificial Intelligence (AI) techniques to an image sequence, that is taken of an environment, in order to analyse and assess what is happening in that environment. In addition, Gonzalez and Woods [47] describe IU as the high-level process of "making sense" of the information

extracted from image sequences that are taken of an environment. IU is usually linked with computer vision.

Image Understanding vs. Image Processing

According to Gonzalez and Woods [47] there is no clear-cut boundary between IU and image processing with the greatest overlap in the area of segmentation and recognition of regions in an image. However, Gonzalez and Woods [47] state that image processing can be considered as a low-level operation that works on input images and produces output images (e.g. the noise removal, image sharpening and enhancement procedures) whereas image understanding can be considered as a high-level operation that works on input images and extracts logical information out (segmentation, classification and recognition, action and behaviour analysis, etc).

Image Understanding Process

The IU process involves a combination of image processing and artificial intelligence tasks that are performed on environment data in order to get an understanding of what is happening in that environment. Once the environment has been captured in a sequence of images the basic back-bone of the IU process is shown below and each part is discussed in detail afterwards. Figure 1 below shows the back-bone of a general IU process.

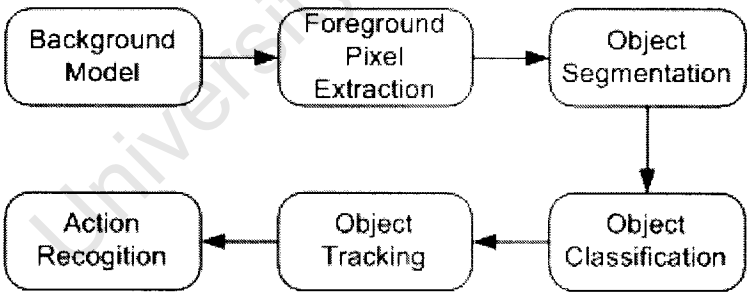


Figure 1: The back-bone of a general IU process

2.1.2 Background Model

Almost all visual surveillance systems have some form of background model (BM) at the heart of their image understanding component(s). The purpose of the BM is to use the data extracted from an input device to represent what the environment looks like without any

foreground objects, hence forming a basis for comparison with the current state of the environment. Usually this comparison leads to the extraction of certain regions of interest, better known as *foreground objects*. Figure 2 below shows a BM image with no foreground objects present on the left hand side and an image with a foreground object present on the right hand side.

Figure 2: A background model without and with a foreground object

In literature, there are many methods that are proposed for constructing the BM. These generally fall into two main categories, namely *non-adaptive* and *adaptive*. The most commonly used BM techniques in literature are presented below.

Non-Adaptive Technique(s)

Non-adaptive techniques generally assume that the environment will not change much and thus construct the BM at initialization of the system without any further changes to the BM during run-time. An example of this would be an average of the first 50 frames of an image sequence being used as the BM. Such techniques are easy to implement and are not computationally intensive. Hence, they may be a good starting point for the implementation of an IU component and may also give the developer an idea of how the environment is behaving.

However, if the environment is constantly changing, e.g. illumination changes, moving trees in the background etc, errors will become evident as time progresses and there will be a need for either a re-initialization of the BM or a more complex BM. According to [4] and [5], non-adaptive techniques would be most suited in "highly-supervised, short-term, tracking applications without significant changes in the scene". Furthermore, according to [4], "Most

researchers have abandoned non-adaptive methods of back-grounding because of the need for manual initialization."

Adaptive Technique(s)

Adaptive techniques assume that the environment is constantly changing and attempt to model the BM to represent those changes. The BM may be constructed at initialization of the system, but then it is always modified or adjusted to adapt to changes in the environment.

1 Previous Image BM

In this type of BM the previous frame is used as the BM and is always replaced by the next image. It is very easy to implement and requires minimal memory and computational resources. This type of BM is mainly used in temporal-differencing which is discussed later in the Chapter. However, the method has several disadvantages which are also discussed later in the Chapter. This type of adaptive BM can also be used as a good starting point for an IU component implementation and may give the developer a bit more clarity on how the monitored environment is behaving.

2 Long-Term Average BM

According to [1] and [5], the most basic adaptive technique is keeping a long-term average of the images in the video sequence which is subsequently and continuously updated. An example of this would be the equation presented in [1]:

$$B(x, y, t) = \frac{1}{t}\sum_{t'=1}^{t} I(x, y, t') \tag{2.1}$$

where $I(x, y, t')$ is the current pixel value for pixel (x, y) at time t' and B is the current BM.

According to [1], a more incremental version of this equation, in which the current pixel value contributes only a certain portion of the BM, is shown on the following page:

$$B(x, y, t) = \frac{(t-1)}{t} B(x, y, t-1) + \frac{1}{t} I(x, y, t) \tag{2.2}$$

The major downfall of this type of incremental equation is the need for storage space to keep the previous *t-1* pixel values. According to [1] an alternative, exponential version of the long-term average would eliminate those storage requirements and would allow the most recent pixel value to have an α weight, the forgetting constant, in calculating the BM. Jacobs and Pless [20] provide a discussion on temporal decomposition using the exponential equation. The exponential equation is shown below:

$$B(x, y, t) = (1 - \alpha)B(x, y, t - 1) + \alpha I(x, y, t) \tag{2.3}$$

According to Friedman *et al.* [1] using exponential forgetting to minimize the effect of illumination variation is equivalent to using a Kalman filter [34], such as the implementation used by Koller *et al.* [29]. In W4 [35], an adaptation of this BM was used, where a combination of the minimum pixel value, maximum pixel value and maximum inter-frame difference value between two consecutive frames are kept for each pixel. These are also updated accordingly when needed.

3 Gaussian BM

In Pfinder [15], the BM was constructed using a Gaussian Model to represent the way each pixel varied over time. This was a novel approach to modelling the BM. This method became the basis for the BM in several video surveillance applications including [1], [4] and [14]. The Gaussian Model used in [15] is similar to the equation shown below:

$$Pr\,(p) = \frac{e^{\left(-\frac{1}{2}(p-\mu)^T \Sigma^{-1}(p-\mu)\right)}}{(2\pi)^{\frac{1}{2}}\,|\sum|^{\frac{1}{2}}} \tag{2.4}$$

Where p is the current pixel values; $Pr(p)$ is the probability of p; μ is the mean; $\sum$ is the co-variance matrix for each pixel; $(\ )^T$ is the matrix transpose and $(\ )^{-1}$ is the matrix inverse.

In literature the co-variance matrix is usually denoted as $\sum$. Some researchers make the assumption that there is independence between the colour channels of pixels in order to simplify the co-variance matrix as shown below:

$$\sum = \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_g^2 & 0 \\ 0 & 0 & \sigma_b^2 \end{bmatrix}$$, where $\sum$ is the co-variance matrix and σ^2 is the variance

In order to construct a Gaussian BM there has to be an initialization period, during which the mean and variance values are calculated for each pixel. These values represent the Gaussian Model that best describes that particular pixel while forming part of the background. After the initialization period, subsequent pixel values can be checked against the Gaussian BM, using metrics which are discussed later, to determine whether they fall into the BM range. Normally, the Gaussian BM pixel-wise mean and variance values are updated to represent the current state of the environment using some form of exponential equation. An example of this is the equation used in [15], shown below:

$$\mu_t = \alpha y - (1-\alpha)\mu_{t-1} \tag{2.5}$$

where α is the learning constant

The trick in this type of BM is to get the most appropriate α value to ensure that a stable BM is obtained and to ensure that the foreground objects in the scene do not corrupt the BM. Furthermore, this type of BM is not designed to handle multimodal backgrounds, such as an outdoor scene. It is primarily used in an indoor environment with a uniform light source. In addition, this type of BM may be resource heavy depending on the type of pixel representation chosen.

4 Mixture Of Gaussian BM

Stauffer and Grimson [4, 24] present the need for a BM that can handle multimodal backgrounds such as outdoor environments with lots of moving trees and background clutter or environments were the illumination changes frequently. They suggest that although each pixel forms a Gaussian cluster around some point, this cluster may shift frequently, thus creating a multimodal background situation. Hence, they propose the need for a multimodal BM. To represent a multimodal BM, they model their BM as a mixture of K Gaussian distributions, which is shown below:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}), \tag{2.6}$$

where ω is the weight of each Gaussian and η is a single Gaussian distribution.

In addition, K is determined by available memory and the computational power of the computer used. When updating the BM, each new pixel value is compared to the K Gaussian distributions, using a defined metric, to determine whether it fits into any of the Gaussian ranges. If no match is found, the least likely Gaussian is replaced with the pixel value as the mean, an initial high variance and low weight. If a match is found, the exponential forgetting equation, described previously, is used to update the mean and variance of the Gaussian that matched. However, by using K Gaussians this type of BM is more resource intensive than a single Gaussian BM. Several other researchers have used this method to represent their BM, including [2], [10] and [30]. Friedman and Russell [1] adapted this type of BM technique by using an Expectation Maximization (EM) algorithm to learn the model.

5 Optical Flow BM

Optical flow is the distribution of motion about a point by an object from an observer's point of view. Hence, in this type of BM the motion of an object is used to model the BM instead of pixel colour. Sidenbladh [6] used the following optical flow (U) in his implementation: U = [u, v], where u is the horizontal flow and v is the vertical flow between a pair of consecutive images in a sequence. In addition, Sidenbladh [6] used a Support Vector Machine (SVM) to train his optical flow BM.

BM Considerations

Although there are a wide variety of techniques available for constructing the BM for a particular system, some consideration needs to be given to the complexity vs. resources ratio that each BM technique presents. For instance, non-adaptive techniques are easy to implement and do not require lots of resources as compared to adaptive Gaussian models, but they do not model the environment as well as a Gaussian model. In other words, one has to ask the question, is the better environment representation worth the resource cost?

Although the effect of illumination variation becomes evident after a foreground pixel extraction method is completed, careful consideration needs to be given to it when developing the BM. If an indoor scene is being monitored that has a static light source, e.g. central light bulb, then a non-adaptive or single Gaussian BM will suffice. However, if an outdoor scene is being modelled were cloud cover can change the illumination of the environment drastically; then a more complex mixture of Gaussian BM needs to be implemented.

Both the benefits and drawbacks of each BM technique and the environment conditions need to be balanced out in order to make a final decision on which technique is most suitable to the environment that is being dealt with.

2.1.3 Foreground Pixel Extraction

Foreground Pixel Extraction (FPE) refers to the process of extracting pixels that are not part of the BM from an image that is being processed. These pixels serve as a basis for further analysis in the following steps of IU. There are several approaches to FPE that researchers use, a few of which are presented below.

Comparison Methods

Comparison methods extract foreground pixels from the current image by obtaining the difference between the current image and a reference image. The reference image is some form of BM. The BM may be as simple as the previous image or have a very complex implementation.

1 Temporal Differencing

Temporal Differencing (TD) refers to the most basic type of comparison operation in which the difference between two or more consecutive frames is extracted to represent the foreground pixels present in the current image. Lipton *et al.* [7] use the following TD equation in their implementation:

$$\Delta_n = |I_n - I_{n-1}| \tag{2.7}$$

where, *In* is the intensity of the nth frame.

TD generally does a good job of extracting moving foreground pixels and responds relatively well to gradual changes in illumination of the environment. However, the major disadvantage of TD is that it does not extract foreground pixels that are not moving. In other words, if a person is stationary in several consecutive frames there is a high chance that the person will not be picked up by TD. As a result, TD sometimes leaves holes in groups of foreground pixels (foreground objects). In addition, TD extracts the shadows of moving objects which, if not handled accordingly, can lead to false analysis in the following IU steps, as explained by

Friedman *et al.* [1]. This technique is also used by Wang *et al.* [9] to extract foreground pixels.

2 Background Subtraction

Background subtraction refers to the process of subtracting the current image from a reference BM representation image in a sequential pixel by pixel manner. This method is commonly used in literature, however according to [5] this technique is highly sensitive to environmental changes, such as illumination variance, camera displacement, background clutter, etc. Thus, this method is dependent on a good BM that models the way the environment changes very well.

The subtraction metric used in background subtraction is dependent on the BM used and the type of pixel colour format used. For instance, when using greyscale values a straight forward subtraction is the most commonly used technique in literature. However, when an RGB, HSV or YUV colour space is used, the subtraction metric may either be the absolute Mahalanobis Distance [27] or the Euclidean Distance [61].

3 Gaussian BM Subtraction

Gaussian BM subtraction is a special case of background subtraction where the BM is a Gaussian BM. In this type of background subtraction the Gaussian Empirical Rule is used to determine whether the current pixel falls into the BM range or is a foreground pixel. A common metric used in literature to determine this condition is to check whether the current pixel falls within 2.5 standard deviations from the mean of the Gaussian BM. If the current pixel falls within that range it is considered as part of the background, otherwise it is considered to be a foreground pixel.

In order to perform a comparison with the Gaussian BM it needs to be converted into a standard normal distribution with $\mu = 0$ and $\sigma = 1$. This is done by converting the random variable that is being calculated into a standard normal variable. Keller and Warrack [28] define the transformation into the standard normal variable (Z) as:

$$Z = \frac{X - \mu}{\sigma} \tag{2.8}$$

Where X is the current value, μ is the mean and σ is the standard deviation.

Hence, an efficient way to determine whether or not the current pixel falls into the 2.5 standard deviation range metric, which was discussed on the previous page, has been presented. If the Gaussian BM was constructed using greyscale values, then a simple subtraction is sufficient for the comparison metric. However, if a multi-dimensional colour space, such as RGB, HSV or YUV was used, then the Mahalanobis distance [27] would better suit as the comparison metric. In addition, Gaussian BM subtraction allows each pixel to have its own range of allowed values which more accurately models the environment than having a global threshold. This type of background subtracting has been implemented in several papers, including [1], [4], [14] and [15].

Optical Flow

In optical flow subtraction, the motion distribution of the pixel is used to extract foreground pixels instead of the pixel colour. This motion distribution is caused by moving objects that are present in the environment. Each pixel's optical flow is compared to a trained optical flow BM in order to determine whether the pixel belongs to the BM or it is a foreground pixel.

The use of optical flow enables one to distinguish between objects which are moving at different speeds. In addition, according to [6], the use of motion instead of pixel colours reduces the effect of camouflage.

FPE Considerations

There are several points to consider when creating a FPE method; some of these are presented below:

1. Representation Of Foreground Pixels

Most FPE methods try to present the extracted foreground pixels in a format that is easily distinguishable from BM pixels, e.g. a unique colour for all foreground pixels, to facilitate the next stages of UI. A common method for doing this in literature is the application of some form of threshold to the foreground pixels so that they can be distinguished from the BM. If the foreground pixel value is above the threshold it is assigned a unique colour (usually white), whereas if it is below the threshold it is assumed that the pixel falls into the

BM range and is assigned a different unique colour to represent that (usually black). This type of resultant image is called a binary image because it only has 2 colours.

A major problem involved with using a threshold is determining what the best threshold for a given environment is. If the threshold is too low, there will be lots of noise mixed in with the foreground pixels. If the threshold is too low then not all the foreground pixels will be extracted. These two scenarios are covered in the object segmentation section.

From a video surveillance perspective, researchers believe that it is more important to get more noise and most of the foreground pixels as opposed to less noise and less foreground pixels. In addition, the noise that is present can be removed by noise removal techniques. In that way the chance of missing a foreground pixel or collection of pixels is minimized.

Another disadvantage of using a threshold method is that normally a single global threshold is assumed. This may not model the environment very well because some pixels may have a wider variation range than others, due to environmental factors such as illumination and background clutter. In such a case a Gaussian BM subtraction method should be used. This would allow each pixel in the BM to have its own variation range. A threshold on all foreground pixels should still be applied to ensure a unique colour for the foreground pixels.

The major advantage of using a threshold method is that it may reduce the processing that is needed in later stages of UI because processing can be restricted to the foreground pixels only instead of the whole image. Some of the papers that used a threshold method are [1], [4], [6], [7], [14] and [15]. For further information about applying a threshold to an image one can read [25].

2. Image Noise

Image noise can be considered as an unexpected or erratic change in a pixel's appearance usually presenting itself as either a min (0) or max pixel colour value (255). According to Russ [25], image noise normally originates from the digital input device. Image noise is noticed when consecutive frames are compared or when a BM is used for comparison. In general, image noise is an undesired phenomenon because it can be detected as valid foreground pixels by the FPE part of the IU system which may ultimately lead to false

assumptions in the object segmentation part of the IU system. Figure 3 below, demonstrates the appearance of random image noise. Two successive images are shown on the left hand side and the difference between the images is shown on the right hand side. This difference demonstrates the random noise that is present.

Figure 3: Random image noise in two successive images

Hence, image noise needs to be dealt with accordingly in the FPE part of the system. Removal of image noise is a well researched area and includes the use of thresholding, Gaussian functions [25], average of neighbouring pixels [25], median filters [25] and morphology [25] and [47]. In [9] removal of noise was accomplished by dividing the binary image into small blocks. In each block the number of foreground pixels was counted. If the number of foreground pixels was greater than a certain threshold, then the block was assumed to be part of the foreground otherwise it was assumed to be part of the BM.

For an in-depth look at the removal of noise from images and morphology [25] and [47] are recommended.

3. Ghosts

In this dissertation the term ghosts refers to the appearance of foreground regions where there is no foreground object present. An effect often experienced when using TD as the background subtraction algorithm. This effect usually appears when a foreground region is present where a person is in the current frame and where the person was in the previous frame. The foreground region where the person was in the previous frame should not be detected at all, but when using a BM that is not sophisticated such effects can happen. This effect is highlighted by Stauffer [4] and Cucchiara *et al.* [33]. In addition, Cucchiara *et al.* [33] propose the use of optical flow to detect and remove ghosts.

4. Removing or Adding Objects to the Environment (Dynamic Scenes)

When building a FPE method, one must always think of the environment that the method is being developed for and how the results of the FPE procedure will affect the BM. For instance, some controlled experimental environments will always have the same objects in their background thus their BM and FPE implementations will be fairly straightforward. However, some environments, e.g. shopping malls, airports, school yards etc, may have constant addition and removal of objects from the background that need to be reflected in the BM. Failure to do this will result in false foreground object detection. An example of this would be a cupboard placed into a school classroom or a poster being removed from an airport building.

A method often found in literature is incorporating foreground pixels that have been in the foreground state for a long period (several frames) into the BM. A few researchers argue that such an implementation can lead to security risks and have presented approaches where several BMs are kept at once and the original BM state will not be corrupted by the temporary addition or removal of foreground objects. Surveillance systems that are based on a multiple Gaussian BM approach, e.g. [2], [4], [10] and [30], tend to have several background layers to cope with objects being introduced and removed from the environment.

The IEEE held a workshop in 2006, called Performance and Evaluation of Tracking and Surveillance (PETS) [41], which were designed to detect abandoned baggage at crowded places, such as airports and railway stations. These workshops have attracted the interest of several researches including: Grabner *et al.* [43], Krahnstoever *et al.* [44] and Smith *et al.* [45].

5. Shadows

According to Friedman *et al.* [1], shadows are one of the most serious problems experienced by video surveillance systems resulting in systems under-counting or over-counting the number of foreground objects by as much as 50%. In an experimental setup, a light source can be placed above the environment to minimize the effect of shadows. However, in a real world environment shadows cannot be dealt with in such a simplistic manner.

In an attempt to remove shadows, Friedman *et al.* [1] model their BM to have 3 classes that a given pixel can be classified as; of which one is a pixel's appearance when a shadow is present. Baba *et al.* [32] present a shadow removal technique based on the density of the shadow. Finlayson *et al.* [31] present a shadow removal method based on extracting a 1-D illumination invariant image from the original image, subtracting it from the original image leaving only the shadows and then using an edge detector to remove the shadow edges.

Some researchers use a much simpler illumination colour space, e.g. YUV or HSV, to extract the illumination component for each pixel in order to detect shadows. An example of this would be the way Pfinder [15] and Cucchiara *et al.* [33] use illumination information to addresses the problem of shadows.

6. Camouflage

In this dissertation camouflage refers to the detection of a foreground pixel as a valid BM pixel due to the similarity in colour of the foreground pixel and its equivalent BM pixel. Camouflage may lead to several problems in colour based BM and may cause holes to appear in groups of foreground pixels (foreground objects). This is further discussed by Harville *et al.* [30]. In addition, Harville *et al.* [30] suggest that to combat the effect of camouflage, a camera that measures depth in the scene needs to be combined with a normal camera when processing the environment.

7. Holes in Foreground Regions

Holes, or regions of background, can appear in the middle of regions of foreground pixels when camouflage is experienced or when the BM model used is not sophisticated enough, e.g. a simple 2 frame TD. In addition, holes can appear in systems where the IU model expects foreground objects to experience a substantial displacement in each frame. This is highlighted by Stauffer *et al.* [4]. If these holes are not dealt with properly the foreground objects may not be segmented correctly. One has to be aware of this possibility and accommodate it in the object segmentation phase of UI.

2.1.4 Object Segmentation

Object segmentation is the process of grouping similar foreground pixels into homogenous regions, better known as *foreground objects or blobs*. The similarity of the foreground pixels is determined by using a similarity metric. In some cases foreground objects may be made up of several segmented regions. The blobs are then processed into the most suitable appearance for the following steps of IU, namely classification, tracking and action recognition.

Similarity Metrics

Similarity metrics are used to determine whether the pixels being compared belong to the same blob and to group the pixels into homogenous regions where all pixels have similar characteristics. There are numerous similarity metrics used for object segmentation that are found in literature, some of which are presented below.

1. Colour Based

In colour based segmentation the similarity in colour of the pixels is used to group the pixels into regions. A common trend is to use neighbouring pixels when checking for similarity in colour. However, some implementations group similar coloured pixels together regardless of the location of the pixels in the environment to form higher level objects. One thing to note in this type of similarity metric is that under varying light conditions the appearance of a colour can change drastically, hence pixels that should belong to the same region may not be segmented together. A Gaussian model for the colour appearance, such as the one used in Pfinder [15], allows for slight variation in the appearance of the colour.

One disadvantage of colour based metrics for object segmentation is that the size and shape of the objects are ignored, which could lead to incorrect segmentation.

2. Proximity Or Location Based

Proximity based metrics focus on the closeness of pixels to determine whether they fall into the same region regardless of the other pixel or object characteristics, such as colour, size. etc. As a result most applications that implement a proximity similarity metric use a binary foreground pixel image in order to remove the redundant pixel information that this metric disregards.

3. Mixture Of Characteristics

Several pixel and object characteristics can be combined in order to get the most accurate representation of what the object looks like and in order to get the most accurate similarity metric. A common method used in literature to accomplish this is to use a model made up of the object characteristics that are combined to form the best representation of that object or region. However, by combining several characteristics together more processing power will be needed to determine similar pixels or regions.

Segmentation Methods

Segmentation methods are usually divided into bottom-up and top-down techniques. Some of these methods are presented below.

1. Thresholding

The simplest bottom-up method used for segmenting objects in an image is thresholding [25] and [47]. This method is based on the idea that regions within an image have relatively consistent colour intensities and can be isolated based on those colour intensities. When thresholding is applied to an image the pixels whose colour intensities fall within the specified threshold range will not be affected, whilst the pixels that do not fall within the threshold will be removed from the image. This process usually results in a binary image. Russ [25] and Gonzalez and Woods [47] define the thresholding operation as:

$$T(x,y) = \begin{cases} 1 & l < p(x,y) \leq u \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

Where $p(x,y)$ is the pixel at co-ordinates x and y in an image, l is the lower threshold limit, u is the upper threshold limit and $T(x,y)$ is the threshold value of the pixel.

The remaining pixels can then be grouped into regions. However, for thresholding to work the exact range of intensities that represents each region needs to be known before hand and each region needs to have a distinct colour intensity range, otherwise merging of regions may be experienced. In addition, if a threshold range that is too large is selected then although the foreground object will be segmented there will be lots of noise in the image. On the other hand if a small threshold range is used then the noise will be less but the foreground object

may not be segmented correctly. A balance has to be reached between noise and segmentation quality. These points are highlighted in the images below:

Figure 4: The effects of various threshold values

Referring to Figure 4 above, the top left picture shows the foreground object that is present in the environment. The top right picture shows the use of a threshold range that is too wide and the resultant noise regions. One thing to note is that although there is lots of noise present in the image the foreground object is almost completely extracted. The bottom left picture shows the use of a threshold range that is too narrow. It is clear that all the noise has been removed but the foreground object has also been removed and is not been segmented correctly. The bottom right picture shows the use of a more balanced threshold range with a morphological noise removal technique to remove the noise. Although the entire foreground object is not extracted, merging techniques can be used to combine the small blobs that make up the foreground object so that it can be correctly segmented.

Russ [25] suggest the use of a two-dimensional threshold, e.g. a threshold for the hue value and a threshold for the saturation value in an HSV colour space, can have better segmentation

success that a single dimensional threshold. On the negative side, the major disadvantage of thresholding is that spatial information is ignored.

2. *Region Growing*

Region growing is one of the most common segmentation methods used by researchers. It is a bottom-up segmentation technique. A common way of implementing a region growing method is to use the idea of starting points or *seeds* which are then grown into regions. Initially, several seeds are determined as starting point. Each seed represents an individual region. These regions are then expanded by adding to them similar pixels determined by a similarity metric. Usually the method stops when all the pixels have been added to at least one region. However, according to [47], in some cases determining the actual point to stop the method may be difficult.

3. *Connected Components*

Connected component labelling is a fundamental operation in most IU systems. It can be viewed as a process where the pixels in the image are processed sequentially and are segmented into similar regions and assigned labels based on their connectivity [59]. According to [59], connectivity means that pixels share the same intensity values and are adjacent to each other. Commonly 4-connectivity, four pixels have to be touching forming a cross with the pixel in question at the centre, or 8-connectivity, eight pixels have to be touching forming a square around the pixel in question, is used. The labels are then used to identify the regions. An example of connected components labelling is shown in Figure 5 on below:

Figure 5: Connected components labelling [59]

From Figure 5 above, one can clearly see the separately coloured components that were correctly segmented using connected components labelling [59]. Several researchers have

used connected component labelling including Lipton *et al.* [7] and W^4 [35]. An alternative to this method is to apply connected component labelling twice in order to maximize the chances of correctly segmenting regions such as the method used by Stauffer and Grimson [4].

4. Region Splitting and Merging

The classic top-down segmentation method is region splitting and merging. In this method, the image is divided into several starting sections which are then combined by a similarity metrics in order to form homogenous regions. This process continues until all the regions are correctly segmented. According to Russ [25], this type of segmentation depends on the segmentation metric used to determine pixels that are not homogenous in each region.

5. Model For Foreground Objects

A model of what the foreground object should look like can be made and used as a reference point to determine whether a particular pixel belongs to that model. An example of this would be the model used in Pfinder [14]. A Gaussian model was used to determine whether or not a particular pixel belongs to a specific blob. The Gaussian model comprises of both colour based and proximity statistics and a covariance between the two groups of statistics.

Object Representation

The representation of a blob is very important for the classification and tracking steps in IU. The representation of the blob refers to the visual appearance of the blob when displayed by the surveillance system to the end user. Quite often there is no need to display all the pixels that form the blob to the end user; hence a more appropriate representation of the blob is generally used. In addition, the representation of the blob affects the amount of memory that is used to store and display the blob to the end user. The process of compacting the blob representation usually result in less processing needed in the classification and tracking steps of IU.

Furthermore, in situations where personal privacy issues need to be protected object representation may be a very important factor. Tansuriyavong *et al.* [36] implemented a monitoring system in which the visual representation of each person on the surveillance system was dependant on the person's preference, so that individuals who needed privacy

protection would not be shown in the video sequences. The most common visual object representations found in literature are silhouettes and bounding boxes.

1. Silhouettes

Object silhouettes are used when the shape or posture of the object needs to be analysed. This type of analysis can be used in both object classification and action recognition. Generally an edge detection algorithm, e.g. Sobel [25], Kirch [25] or Canny [25] edge detection operators, is applied to a foreground region in order to extract the silhouette of the region. Edge detection is discussed in some detail by Russ [25] and Gonzalez and Woods [47]. In [36] silhouettes are used to hide a true person's appearance to protect the person's privacy. In [38] and [45] silhouettes are used to outline the people that are tracked. An example of the use of silhouettes to represent foreground objects is shown in 6, below:

Figure 6: Showing the use of silhouettes to represent foreground objects extracted from [45]

2. Bounding Box

A bounding box that encapsulates all the pixels in a region is the most common technique found in literature that is used as the representation of the blob. A bounding box can give the system measures for the height, width and centre of the blob which can be used in object classification and tracking. Examples of the use of bounding boxes are [2], [16], [35] and [37]. One problem with using a bounding box to model the blob is that the blob's posture information will not be available. The use of a bounding box to represent the foreground objects is shown in Figure 7 on the next page:

Figure 7: The use of bounding boxes to
represent foreground objects extracted from [43]

3. Other Methods

In some instances where only people are tracked, an ellipse is used instead of a bounding box as the shape of the blob because it models the shape of a human better that a bounding box. If the objects that are being tracked are small, e.g. people at a distance away from the camera or birds in the sky, the object may be represented as a single point (typically the centroid) but such representation will not allow the system to use shape or posture information. Usually such representations are used when speed characteristics are needed for classification and tracking.

In Pfinder [15] the people in the scene are represented as multiple connected blobs, i.e. the head, body, hands and feet are all different connected blobs. Such a multi-blob representation is very useful when hand gestures or body posture information is required, but may be more computationally expensive than a bounding box approach when tracking is performed.

Object Segmentation Considerations

A major problem in object segmentation is the incorrect extraction of foreground regions. The main causes for the extraction of incorrect foreground regions are the appearance of noise and different appearances of the object in the image due to occlusion, e.g. a person standing behind a desk may not be segmented correctly. Due to the fact that noise regions are random and usually non persistent events, such noise regions can be handled efficiently by

implementing a segmentation algorithm that waits for several frames to see if the blob persists before segmenting it.

Hence, one can clearly see that object segmentation is highly dependent on a good foreground pixel extraction mechanism and a good noise removal method. Likewise, the rest of the IU processes are highly dependent on the correct segmentation of the blobs that appear in the scene. For an in depth discussion of object segmentation, the Image Processing Handbook [25] and Digital Image Processing [47] are recommended.

2.1.5 Object Classification

Object classification is the process of identifying what kind of object is present in the environment. This is particularly useful when distinctly different types of objects are present in the environment and when different tracking methods are used for the various objects. Some examples of applications that track varying objects are [7], [19] and [38], where cars and people are monitored. Another example would be differentiating between a single person and a group of people such as in school environments or at sports matches. Ribeiro *et al.* [23] also differentiate group activity from individual behaviour.

According to [5], object classification can be considered as a standard pattern recognition process. Hence a pattern that best represents the object is determined and used to identify the object in the scene. Furthermore, the classification rules that are used should be computationally inexpensive yet sophisticated enough to accurately classify objects in an environment.

Classification Metrics

There are various metrics found in literature for object classification, a few of which are presented below.

1. Size Metric

Size can be used as a classification metric when the objects in the environment have distinctly different sizes. For example, in [43], [44] and [45] the size metric is used to differentiate between humans and luggage that is left in crowded places such as airports and

railway stations. An example of the use of a size metric for classification is shown in Figure 8 below:

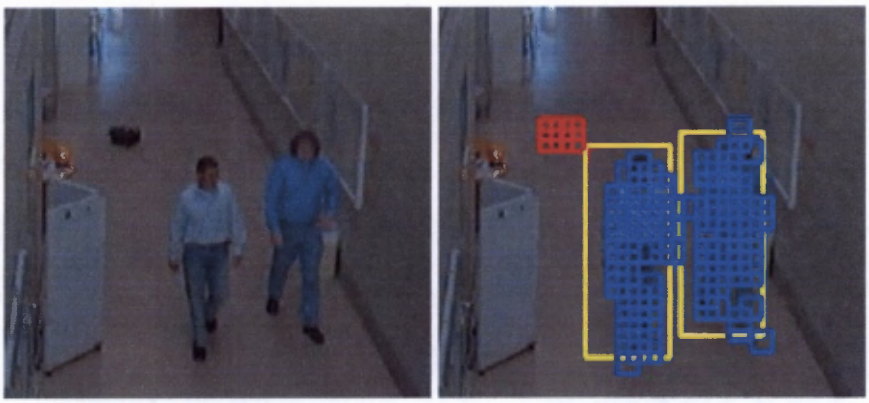


Figure 8: Different sized objects [43]

From Figure 8 above, one can see that the larger individuals in the environment are coloured in blue whereas the small abandoned baggage is coloured red. The problem in using this metric becomes apparent when the objects that need to be classified are of similar size or when occlusion happens resulting in the change of an object appearance. In such circumstances misclassification is common.

2. Speed Metric

The speed of an object can also be used to classify different objects. This type of metric should be used in an environment where the objects have distinct speed classes. An example of the use of speed to classify objects is the implementation in [2] where a speed metric was used to classify people on foot and people riding on bicycles. Again the problem becomes evident when the speeds of the objects become similar and misclassification happens.

3. Dispersedness

Dispersedness is a ratio that can be used as a classification metric to differentiate between different sized objects. Lipton *et al.* [7] used dispersedness to differentiate between cars and humans and the ratio calculation that they used is shown below:

$$Dispersedness = \frac{Perimeter^2}{Area} \tag{2.10}$$

Having prior knowledge that humans are generally smaller than cars, Lipton *et al.* [7] were able to obtain unique dispersedness ratios for humans and for cars which made classification easier. The ratios used by Lipton *et al.* [7] are shown in Figure 9 below:

dispersedness : 61.8 dispersedness : 41.0

Figure 9: The dispersedness ratios used in [7]

Object Classification Considerations

When deciding on a classification algorithm, the first thing to think about is how many objects are present in the environment. If only one object is present then a very simple classification algorithm can be formulated. However, if there are several objects that are present in the environment then a distinguishing characteristic amongst the objects needs to be identified. Sometimes when the objects are very similar more than one distinguishing characteristic needs to be considered. One has to keep in mind that the more characteristics that are considered in the classification algorithm the more processing time it will take and this has to be balanced with the classification needs of the surveillance system.

2.1.6 Object Tracking

The purpose of tracking a foreground object or "blob" in an environment is to know what kind of actions the blob is doing so that a higher level understanding of what is going on in the environment can be achieved. In addition, by tracking the blob's actions a prediction of the next possible action can be approximated based on the history of the blob's actions. Object tracking is highly dependent on a good preceding object segmentation step. Furthermore, in most IU systems object tracking and action recognition are run in parallel and are dependent on each other. Typically the tracking data that is obtained during object tracking is processed by the action recognition system that determines what action the blob is doing, and that action is then used by the object tracking system to match the blob in the next frame.

Object Matching Techniques

The first and most critical step in object tracking is to make sure that the same blob is being tracked in each subsequent frame. This is a very important step in many video surveillance applications because for correct tracking of blobs the actions of different blobs in the environment should never be mixed up. The next step is dependent on the implementation of the action recognition system, but usually involves getting the location of the blob and the distance moved from the last frame. There are several approaches to object matching in literature, a few of which a presented below.

1. Proximity Based

Proximity based matching techniques match blobs in consecutive frames by the inter-frame displacement that is experienced by the blob. Usually a bounding box centroid is worked out for each blob and used as the reference point for measuring the blob's displacement. The displacement of the bounding box centroid is then measured using the Euclidean distance [61] between consecutive frames. If this displacement is below a certain threshold then it is assumed that the same blob has been matched and tracking can continue. Examples of the use of a bounding box centroid are [2] and [16]. Other researchers use different blob characteristics, such as direction or speed, to make sure that it is in fact the same blob that has been matched.

2. Prediction Techniques

In some video surveillance applications prediction techniques are used to predict a blob's characteristics in the next frame and matching is done according to those predicted characteristics. A few examples of the characteristics that can be predicted are speed, direction and location. A Kalman filter [34] can be used to predict an object's location or speed in the next frame. Several researchers, including [2], [24] and [46] used a Kalman filter to match the objects that were being tracked in their environments. W^4 [35] and the work of Senior [37] also used a prediction technique to estimate where the centroid of their blob would be located in subsequent frames in order to match and track blobs.

3. Using Blob's Characteristics

As mentioned previously, several other blob characteristics besides the blobs location can be used for matching and tracking purposes. For example the size, speed or direction of movement can be used to match blobs in consecutive frames. The main problem of this approach is that the blob's characteristics can change from frame to frame. For example, if a person is moving away from the camera the blob will become smaller and smaller and may not be matched correctly. Thus, a more reliable metric such as blob location should be used in conjunction with the blob's characteristics to match a blob correctly. The Euclidean distance [61] can be used to work out a match in a multi-dimensional characteristic space.

4. Blob Model Based

This type of object matching metric uses a blob representation model to make sure that the same blob is being tracked throughout the environment. Usually a Minimum Description Length (MDL) criterion is used for the representation that best models the object that is being tracked. In Pfinder [14] a Gaussian model is made for every new blob that appears in the scene and added to a list of blobs present in the scene. In every frame each blob's model is matched to the list of existing blob models that are present in the scene to determine which blob it is. A Gaussian mixture model is also used by Chen *et al.* [21]. Lipton *et al.* [7] used an appearance based object template in correlation with TD to detect moving objects.

Object Tracking Considerations

The major problem in object tracking is incorrectly matching a blob in consecutive frames, mainly due to occlusion of the blob by other blobs or background objects. The tracking system has to be sophisticated enough to handle the object occlusions as well as to handle the tracking of objects that are close together. Usually that scenario results in the merging of the two close blobs into one blob and a region splitting algorithm needs to be implemented which can then split the two merged blobs.

Another problem would be the re-appearance of the same object into the environment after a few frames have passed. This situation can lead to a new object being detected instead of the same object being matched and thus incorrect tracking may happen. A simple solution that is commonly used is to keep track of objects that have left the environment for several frames, and then when the objects re-appear they will be correctly matched.

2.1.7 Action (Behaviour) Recognition

Action and behaviour recognition have been at the forefront of recent efforts by visual surveillance researchers. The need to understand what is happening in an environment is a vital and critical task; not only in security environments, where restricted areas are monitored such as airports [14] and shopping malls [16], but also in everyday environments such as schools and work places [13]. Further examples of recent action recognition research include: sea port surveillance [39]; nursing homes [21]; abandoned baggage detection at crowded places [43], [44], [45] and [46]; and the work of Stauffer *et al.* [4] in general action recognition.

Initially action recognition systems focused on detecting simple actions based on a blob's tracking data but recently focus has been directed towards determining the behaviour of blobs which may be considered as complex actions that are made up of several simpler actions. Such complex actions could not be detected by analysing tracking data alone but instead required the use of sophisticated action recognition systems.

The main problem with an activity recognition system is incorrect activity recognition as a result of receiving incorrect data from the tracking system. Hence, an action recognition system is highly dependent on a good tracking system that can reliably track a single blob and extract vital data that is then used to determine what kind of action the blob is performing. On the other hand, the action recognition system should have the simplest possible representation of the actions that it can recognize in order to minimize its dependence on the tracking system's ability to extract the vital data. In some instances where prediction is used for tracking, the action recognition and tracking systems have bidirectional dependence.

Types of Actions

The actions that can be recognized by surveillance systems can be divided into two categories namely: *simple* and *complex* actions.

1. Simple Actions

Simple actions are actions that can be determined by direct analysis of a blob's tracking data. There is no sophisticated analysis engine that is needed to understand the simple actions. Researchers have used various types of techniques to determine simple actions that occur in

environments. Ribeiro and Santos-Victor [23] used optical flow of a blob' bounding box centroid to determine the following simple actions: blob being active or inactive, walking, running and fighting. Similarly, Bador *et al.* [2] used the blob's motion to detect simple actions such as walking, running and loitering. In addition, Nascimento *et al.* [16] use a blob's motion patterns, based on the blob's centroid, to detect actions such as exiting a shop, passing or browsing in front of shop windows. Finally, Leo *et al.* [50] use pre classified human posture to recognize the activity that is going on in the environment.

Another popular simple action that can be recognized is entry into a restricted area, as presented by Bador *et al.* [2]. This is determined by analysing the position of the blob in relation to a predetermined restricted area location. Furthermore, in some cases the tracking data can be combined to determine certain simple actions. An example of this would be Bador *et al.* [2] who detect if a person has fallen down on the ground by combining the blob's motion and position tracking data.

2. *Complex Actions*

Complex actions are actions that cannot be determined by a simple analysis of tracking data and require a sophisticated action recognition system. Usually complex actions are made up of several simple actions occurring sequentially. According to Lou *et al.* [11], Neural Networks (NN), Hidden Markov Models (HMM) and Bayesian Networks (BN) are some of the sophisticated statistical models that have been used in literature for complex action recognition.

Detecting interactions between different blobs in an environment is a common complex action recognition task. Ivanov *et al.* [10] use an event generator, which maps object tracks onto a set of pre-determined discrete events, and a stochastic parser to detect several events that occur in a person and car interaction environment. In addition, Park and Trivedi [19] detect interactions between people and cars.

Another example of recognizing complex actions is the detection of abandoned luggage and the person who abandoned the luggage at a crowded place (airports and railway stations) [43], [44] and [45]. This is accomplished by using the size of the blob to detect the left

luggage and the radius distance around the language to detect the person associated with the left luggage. This is shown in Figure 10 below:

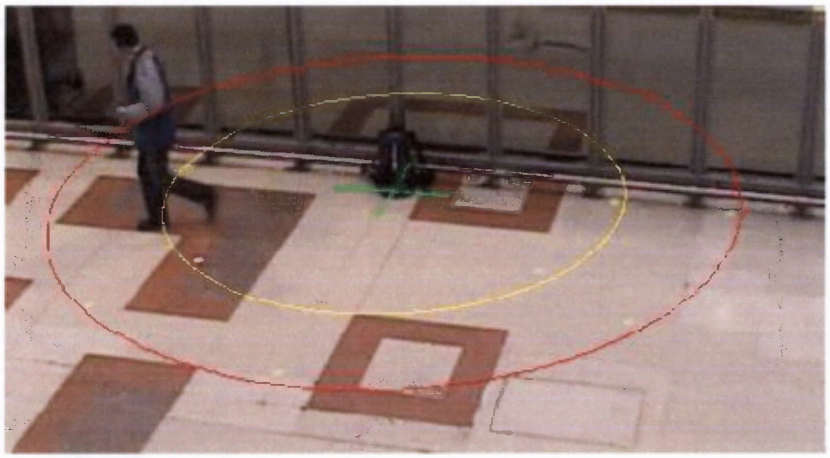


Figure 10: The use of a radius distance to determine who left the abandoned luggage [45]

In Figure 10 above, the yellow circle represents the allowed radial distance that the individual can move away from the location of the luggage without raising an alarm. The red circle represents the alarm radius distance. The distance between the yellow and the red radii is the warning distance. If the individual moves further away from the luggage than the red radius then the luggage can be determined to be abandoned.

Lou *et al.* [11] use trajectory paths and an HMM to detect complex car and person activities. Thirde *et al.* [14] use *a priori* knowledge of the environment and a predetermined set of action event models, based on expert knowledge, to detect actions on an airport apron. Cohen *et al.* [17] use a Bayesian Network to recognize facial expressions. Park and Aggarwal [18] use a Bayesian Network to determine when an interaction occurs between two people. Chen *et al.* [21] use an ontological Bayesian Network approach to analyse social interactions in a nursing home.

2.1.7 Action Prediction

Action prediction is an area that has received insufficient effort by the researching community. Prediction has been used by several researchers including [2], [24], [35] and [37] for predicting the location or speed of an object in the next frame or time step for tracking purposes, but little work has been done on analysing predicted actions to determine whether a

risk may be present. This fact is highlighted by Hu *et al.* [5] who state that action prediction is amongst the list of video surveillance issues that is still open and deserves further research. Duque *et al.* [49] use a Dynamic Oriented Graph (DOG) to predict abnormal object actions in a video sequence. Their implementation is based on object tracks and monitoring of restricted areas.

Action prediction is an extremely difficult thing to do. An example would be the common weather predictions that are incorrect by meteorological departments.

2.2 Artificial Intelligence (AI)

This section presents some of the research techniques in the field of AI that are needed to understand the implementation of the prediction system part of the prototype that was developed during this dissertation.

2.2.1 Artificial Intelligence Overview

The term Artificial Intelligence (AI) is generally considered to have been first coined by John McCarthy in June 1956 [53] and [48]. However, according to Kelly [53] it is quite impossible to capture the full meaning of such a widely inclusive term in a succinct, or compressed, description. In Kelly's discussion of AI he combines the definitions of several researchers in order to get a broad understanding of AI. The discussion touches on several researchers' definitions of AI which are summarized into 3 categories of emphasis. These are listed below:

- Attempting to make machines smarter
- Modelling activities of human intelligence
- Exploring the position of computation in the space of possible intelligent entities

Russell and Norvig [48] concur with Kelly's opinion that there are several views on AI by various researchers and they state that the definitions of AI vary in two main directions, which also concur with some of the categories that Kelly specified, namely:

- thought process and reasoning (thinking and acting like humans)
- behaviour (thinking or acting rationally)

In an attempt to draw an approximate boundary around AI, Rich and Knight [54] define AI as a study of how to make computers do things which, at the moment, people do better.

2.2.2 Probability Theory

Probability theory provides a foundation of methods for dealing with uncertainties that happen in real world environments in a consistent and rational manner [28]. Each action in the real world, or *random experiment* [28], can result in a number of mutually independent outcomes, or *sample space* of *simple events* [28], which needs to be represented in a statistical manner so that certain decisions and assumptions can be made about those outcomes.

Probability

The probability of an outcome or simple event is a measure of how likely it is that the outcome will occur. It is denoted as $P(E)$, where E is the outcome or simple event. A probability of an outcome must follow the following rules, adapted from [28]:

Given a sample space $S = \{E_1, E_2 \ldots E_n\}$

1. $0 \leq P(E_i) \leq 1$ for each i (2.11)
2. $\sum_{i=1}^{n} P(E_i) = 1$ (2.12)

Probabilistic Notation

The following notations are used in probabilistic theory to describe events that occur:

1. P (A) is the probability that an event A occur. (2.13)
2. P (¬A) is the probability that an event A does not occur. (2.14)
3. P (A ∧ B) is the probability of both A and B occurring. (2.15)
4. P (A, ¬B) is the probability that A occurs and B does not occur. (2.16)
5. P (A) + P (¬A) = 1 has to be true for all probabilities. (2.17)

Conditional Probability

Conditional probability makes use of partial knowledge about related, dependant events and is denoted as P (A | B) - the probability of A given B. In other words, the fact that B has

occurred directly affects the probability of A occurring. It is defined by Keller and Warrack [28] as:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} \quad \text{given that P(B)} > 0 \tag{2.18}$$

Similarly,

$$P(B \mid A) = \frac{P(A \wedge B)}{P(A)} \quad \text{given that P(A)} > 0 \tag{2.19}$$

In addition, according to [28], the above conditional probability formulae can be re-written, in order to get a formula of the joint probability P(A ∧ B), in the multiplication or product rule form as:

$$P(A \wedge B) = P(A \mid B)\, P(B) \text{ or } P(A \wedge B) = P(B \mid A)\, P(A) \tag{2.20}$$

Independence of Events

Events are independent if the occurrence of one event does not affect the occurrence of the other event. Keller and Warrack [28] state that two events, A and B, are independent if:

$$P(A \mid B) = P(A) \text{ or } P(B \mid A) = P(B) \tag{2.21}$$

Hence, the conditional probability of independent events can be written as:

$$P(A \wedge B) = P(A) * P(B) \tag{2.22}$$

Bayes' Rule

Bayes' Rule is a formula that allows unknown conditional probabilities (effects) to be computed using know conditional probabilities or evidence probabilities. The formula is based on the combination of the product rule format of the conditional probability formula.

By equating the two forms of the product rule, discussed earlier, one gets:

$$P(B \mid A)\, P(A) = P(A \mid B)\, P(B) \tag{2.23}$$

If both sides of the above equation are divided by P (A) one gets the formula for Bayes' Rule. This formula is shown below:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{2.24}$$

Often in literature Bayes' Rule is expressed in the form presented below:

$$P(H|e) = \frac{P(e|H)P(H)}{P(e)} \tag{2.25}$$

where H represents a hypothesis and e represent the evidence or prior probability.

In addition, *P (H | e)* is called the posterior probability. According to Russell and Norvig [48], the use of Bayes' Rule underlies all modern AI systems for probabilistic inference.

2.2.3 Modelling and Knowledge Representation

Modelling and knowledge representation are AI fields which deal with representing real world scenarios, events and actions in a structured, computational form so that an understanding of the environment can be obtained and some form of reasoning can be performed about the environment by a computer system. One must note that modelling an environment in its entirety is almost impossible [48]. Nevertheless, the most important details about the environment can be extracted and a reasonable, estimated model can be developed that approximates the environment which can then be used to reason about the environment.

Symbolic reasoning may be used to represent a particular environment in the real world, the objects that are present within that environment and the actions performed by the objects in a symbolic form that can be used by a computer system to model the environment. To highlight this fact, Rich and Knight [54] state that the basics of any a knowledge representation model:

- Facts – the objects and events that are present in the environment
- Representation of the facts in some chosen formalism.

In addition, Rich and Knight [54] state that structuring these basic facts can be done in two levels namely the knowledge level, where the facts are specified, and the symbol level, where

the facts are represented as symbols and can be reasoned about. Hence, in the knowledge level the most important facts that represent the environment are extracted and a structure of the model is designed. In the symbol level, the model is transferred into a computational symbol representation.

Once a model has been designed and developed, Rich and Knight [54] argue that probabilistic reasoning about the model can be performed.

2.2.4 Probabilistic Reasoning

Probabilistic reasoning is an AI field devoted to reasoning about the real word models that are usually developed by using some modelling and knowledge representation technique. Probabilistic reasoning assumes that all the events in the model are random events and can be represented in a statistic manner. Furthermore, in order to reason about the real world model Rich and Knight [54] state that some knowledge about how the model works needs to be obtained, usually by collecting a large amount of data, and a mechanism to manipulate that knowledge needs to be designed.

The process of getting knowledge about the real world and representing it in a computational form, usually a statistical form, forms part of a process called learning. This process requires monitoring, analysing and examining the environment or data about the environment in order to calculate the statistical data about the events and objects present in the environment. There are several techniques in AI for learning of data including data mining and data capturing. Once learning has been performed then probabilistic reasoning about the environment can be performed. This process is called inference.

2.2.5 Bayesian Networks (BN)

This section presents a powerful AI modelling and probabilistic reasoning data structure called a Bayesian Network. The majority of the discussion is adopted from [48].

Bayesian Network Overview

A BN is a statistical data structure that uses the AI techniques of modelling, knowledge representation and probabilistic reasoning to represent random events that are present in an

environment and the way that those events are related. The relationship between the events is done in the form of conditional probability based on related data, or prior evidence, about the events. In addition, the relationships between the events in the network are represented as cause-effect relationships and each event is specified using quantitative probability information. However, a BN does not model temporal relations between and within the events. Instead, an extended BN known as a Dynamic Bayesian Network (DBN) should be used to model temporal relationships and trends.

A BN is best represented graphically using a Directed Acyclic Graph (DAG) structure. The BN DAG has to conform to the following rules:

1. A set of random variables, either discrete or continuous, or uncertain quantities make up the nodes of the BN.
2. Each node has a set of states that it can be in.
3. A set of directed links or arrows connects pairs of nodes. This is a representation of a cause and effect relationship.
4. Each node N (or effect node) has a conditional probability distribution P (Ni | Causes (N_i)), which is stored in a conditional probability table (CPT), that quantify the effect of the cause(s), or parent(s), on the node. In addition, the conditional probability distribution is commonly calculated using Bayes' Rule.
5. The graph has no directed cycles.
6. The root(s) of the network should have a prior probability P(root) instead of the CPT

The following example, adopted from [48], gives an example of a BN.

Scenario: A new alarm system has been installed at a person's house and responds when a burglary occurs. However, the alarm also responds when an earthquake occurs. When the alarm goes off, the police and the fire brigade are alerted. Sometimes the police and the fire brigade respond to the notification, other times they do not. This example can be represented as a BN and is shown below in Figure 11 in the BN DAG form.

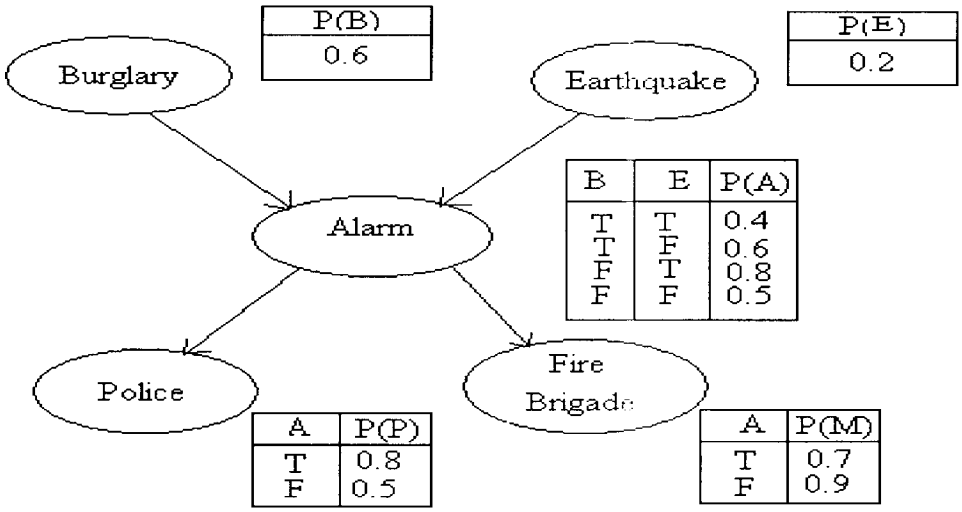


Figure 11: A Bayesian Network

In Figure 11 above, the table next to each node represents the CPT table for that particular node. Each node in this example has only 2 states that it can be in, true or false. P(X) denotes the probability of the node X being in a true state. This could be extended to include the conditional probability of each node being false by the simple calculation 1-P(X). The arrows show the cause and effect relationships between the nodes in the BN. The direction of the arrows shows the direction of causality amongst the nodes.

Hence, from the above example, one can conclude that a BN can be viewed as a joint probability distribution (JPD) [48]. At the same time, a BN can also be viewed as an encoding of CPTs [48]. These views of a BN are equivalent, although according to [48] it is easier to understand a BN by using a JPD. In general, each entry in the CPT table of an effect node represents the probability of the effect node being in that particular state. This probability can be calculated by combining the probabilities of the specific evidence information from its causal nodes' CPT entries that specify it into a JPD.

According to [48] the formula for calculating the value of a general entry in the CPT table of an effect node specified by a JPD of causal node states is:

$$P(Y = y) = P(X_1 = x_1 \wedge X_2 = x_2 \wedge \ldots \wedge X_n = x_n) = \prod_{i=1}^{n} P(x_i | causes(X_i)) \quad (2.26)$$

Where the P (Y=y) is the probability of effect node Y being in state y and P ($X_i = x_i$) is the probability of causal node X_i being in state x_i. This JPD can be extended using the product and chain rules to account for any number of nodes in the BN. An example of the above JPD calculation of an effect node's general state value, using the previously presented BN, would be as follows:

Scenario: The alarm has sounded (a), there was no burglary (b) and there was no earthquake (e), but both the police (p) and the fire brigade (f) have come to the house.

$$P(\mathrm{p} \wedge \mathrm{f} \wedge \mathrm{a} \wedge \neg\mathrm{b} \wedge \neg\mathrm{e}) = P(p \mid a)\, P(f \mid a)\, P(a \mid \neg b\ \neg e)\, P(\neg b)\, P(\neg e)$$
$$= 0.8 \times 0.7 \times 0.5 \times 0.4 \times 0.8 = 0.0896$$

The example on the previous page is an example of the most common application of a BN, namely inference.

Finally, each CPT table represents the possible combinations of the cause and effect nodes. The quantitative information about these combinations is generally obtained by using a number count of the occurrences of the effect, given that the cause has occurred or given that the cause has not occurred. Importantly each row in the CPT must sum up to one. If this is not the case then normalization of the values in the row needs to be performed. However, the column sums of the CPT do not have to add up to one.

For an introduction into BN [58] is recommended and for an in-depth discussion of BN [48] is recommended.

Learning

From the above overview of BNs one can conclude that two things are of critical importance to describing a BN, namely the network structure of the BN and the parameters of each node's CPT table. If the network structure is not correct then the assumptions implied by the causality relationships between the nodes may be misleading and could eventually lead to erroneous conclusions and interpretations about the world model that the BN is modelling. Likewise if the parameters of the CPTs are not correctly specified then similar erroneous

conclusion may be drawn. This can be the case when a person who is not an expert in the observed environment specifies the model structure and CPT parameters.

Murphy [58] suggests that both the structure and CPT parameters can be learnt from existing data. However, learning the structure of the BN can prove to be an NP-hard problem [58]. In other words the problem may prove to require a non deterministic polynomial amount of time to solve. In addition structure learning may be much more difficult than CPT parameter learning, especially when the BN has hidden nodes that are not directly observed from the training data. This fact is highlighted by Murphy [58]. When the structure of the BN is known the learning process is referred to as supervised learning whereas when the network structure is not known the learning process is referred to as unsupervised learning.

Murphy [58] presents the following four possible cases for learning a BN.

1. When the network structure is known and data variables are observable
2. When the network structure is known and the data variables are hidden or missing.
3. When the network structure not known but the data variables are observable
4. When the network structure not known and the data variables are hidden/missing.

1. Known Structure and Observed Data Variables

In the first case where the structure is known and the data variables are fully observable the learning that is required would be parameter learning of the CPT. This situation may arise when an expert in the observed environment knows how the environment objects and events are related and builds the model based on his/her knowledge and uses existing data about the model to train the CPT parameters. [58] and [48] suggest the use of a maximum-likelihood estimation (MLE) algorithm to perform parameter learning in this case, in order to maximize the likelihood of the training data. However, Murphy [58] warns that the MLE algorithm is prone to sparse data problems.

The following example, adopted from [48] is used to illustrate the formation of an MLE algorithm:

Scenario: A producer supplies bags of balls to a sport facility, which can contain either soccer balls or volley balls. The probability distribution of the balls in each bag is not known until the bags are opened. This scenario can be modelled by assuming that the probability of

getting a bag with only volleyballs is θ, and the probability of getting a bag with only soccer balls is $1-\theta$. Now suppose that N bags were received, of which there were s soccer balls and v volleyballs and d is the likelihood data obtained about the type of ball contained in each bag after opening each bag. The task is to predict the fraction of volleyballs in a particular bag which can be denoted a hypothesis prior h. The following mathematical product model can be used to model this particular scenario:

$$P(d|h) = \prod_{j=1}^{N}(d_j \mid h) = \Theta^v(1-\Theta)^s \tag{2.27}$$

The MLE can be used to do parameter learning in this case and is given by the value of θ that maximizes the above expression. According to [48] the same value can be obtained by maximizing the log-likelihood L. This is shown below.

$$L(d|\,h) = \log\big(P(d \mid h)\big) = \sum_{j=1}^{N}\log\big(P(d|h)\big) = v\log\Theta + s\log(1-\Theta) \tag{2.28}$$

The main advantage of using logarithms is that the product formula is reduced to a sum over all the data. According to [48] in order to find the maximum-likelihood value of Θ, L is differentiated with respect to Θ and equated it to 0. This is shown below.

$$\frac{dL(d\,|h)}{d\Theta} = \frac{v}{\Theta} - \frac{s}{1-\Theta} = 0 \Rightarrow \Theta = \frac{v}{v+s} = \frac{v}{N} \tag{2.29}$$

Hence, the proportion of volleyballs in a given bag is the proportion of observed volleyballs from the existing data. Thus from the example, this type of learning is a simple count of occurrences over the entire data set. However, [48] adds that at times the last step in the MLE algorithm may prove to be difficult and may require an iterative evaluation of possibilities to determine the maximum likelihood of a parameter occurring. When the data is continuous, a Gaussian distribution can be used to perform MLE. A more detailed elaboration on the MLE algorithm can be found in [48] and [58]

This case assumes an ideal condition where both the structure of the BN is known and the data variables are observable. This is not usually the case in a real world environment.

2. Known Structure and Missing/Hidden Data Variables

The second case arises when some data is missing or when some events are hidden or not represented by the data. An example of this would be modelling a windy environment and observing the effects of wind but not actually seeing the wind. In that case wind would be a hidden or latent variable. According to [48] and [58] the Expectation Maximization (EM) algorithm should be used in this case to find a locally optimal MLE of the missing parameters or latent variables.

The EM algorithm has two main steps, namely the E and M streps, in its implementation. The first step in the EM algorithm, or the E step, is to work out the expected values of all the missing data and latent nodes by using a querying or inference algorithm. These expected values are then treated as observed data by the second step of the algorithm, or the M step, which performs learning using the normal MLE algorithm to maximize the likelihood of the data. An exhaustive explanation of the EM algorithm can be found in [48].

In addition, according to [48], by using latent variables to represent missing data in a BN the number of parameters required to specify the BN is significantly reduced which in turn may result in less data being needed for the learning the parameters of the BN.

3. Unknown Structure and Observed Data Variables

When all the data is present but the structure of the network that best represents the data is not known, Murphy [58] suggests that a search through all possible network structures needs to be performed in order to determine which network structure fits the data best. Such a scenario may occur when there is no expert in the observed environment that knows how the objects and events in the environment are related. However, Murphy [58] warns that this problem is an NP-hard problem, which means that finding the solution to the problem may take a non-deterministic polynomial amount of time to complete, because the number of networks on N variables is super-exponential in N.

In order to find the best possible network, a scoring metric needs to be specified so that the network structures can be evaluated and compared. The network structure with the maximum score can then be assumed to be the best representation of the data. An example of such a

method is the greedy algorithm. However, one must be weary of local maxima conditions and have some way of detecting them. This case is explained in more detail in [48] and [58]

4. Unknown Structure and Missing/Hidden Data Variables

According to Murphy [58], the last case where both the network structure is not known and the data variables are hidden or data is missing is the hardest case of all. A possible way of solving it would be to use a combination of the EM algorithm for parameter learning and a network structure search to find the best network structure. Murphy [58] also suggests that introducing new hidden variables can make the BN model more compact and can sometimes lead to easier structure learning and better results. However, interpreting the meaning of the newly introduced hidden variables may be problematic. This case shows an example of unsupervised learning.

Inference

According to Russell [48], the basic task of inference in a BN is to establish the posterior probability of the state of a query node given the state of some observed node or evidence node. In other words, this means that we wish to know the probability of an effect node given the current state of the cause node. This type of inference is called top-down inference. In some instances inference about the cause needs to be performed based on the evidence of the effects. This type of inference is called bottom-up inference. In addition, inference in a BN can be either exact or approximate. The choice of inference methods to use is dependent on the network structure and the quantity and availability of data.

In order to perform exact inference, a reasonably sized data set is required. If the data set is too small exact inference may not get accurate probabilities. According to Murphy [58], when the data set is very large, exact inference becomes very slow. Hence if the data that is available is sparse, very small or extremely large then approximate inference would be more suitable if approximate probabilities are sufficient for the task at hand. In addition, according to Russell [48], exact inference is an NP-hard problem and it can be shown that when large multiply connected graphs are used exact inference algorithms prove to be as hard as that of computing the number of satisfying assignments for a proportional logic formula, which makes them #P-hard. In some cases exact inference tends to be more computationally

expensive than approximate inference. Thus computational power vs. accuracy needs to be weighed out when choosing the type of inference to use.

Where exact learning is applicable several algorithms can be used including:

- Enumeration [48]
- Junction tree [58]
- Variable elimination [48] and [58]
- Clustering algorithms [58]
- Linear algebra (for Gaussian nets)
- Pearl's message passing algorithm (for polytrees) [58]

When exact inference is not possible, e.g. in large multiply connected BNs [48], approximate inference is used instead. Murphy [58] suggests that stochastic methods can be used to perform approximate inference. The algorithms used for approximate inference include:

- likelihood weighting [58]
- sampling algorithms, e.g. Markov chain Monte Carlo (MCMC) [58]

All the inference methods are covered extensively by Russell [48] and Murphy [58].

JavaBayes

JavaBayes [57] is a java application developed for modelling and visualizing BNs. The application allows the user to create a BN, specify parameters for each node and assign values to the parameters. In addition, it provides functionality for inference. The application is released under the GNU General Public License which allows free use of the application.

2.2.6 BaBe

BaBe [62] is an adaptive agent framework which is made up of a collection of agent-based software tools that can provide real-time environment modelling functionality by creating a complex adaptive system (CAS) of an environment. A CAS is a system that is made up of individual components. In addition, a CAS can modify its behavior to match the changes in

the environment that it is modeling by learning the patterns in the environment and adapting to them. Most importantly, a CAS exhibits the phenomenon of *emergence*[1].

More specifically, the BaBe framework consists of simple, resource-constrained, adaptive BaBe agents, which provide AI techniques and methods, and BaBe Front Ends, which represent the interfaces to the framework. In addition, Potgieter [62] states that "the BaBe adaptive agents can be used to build self-organizing and self-coordinating systems, which are based on the aspect of communication via modifying the local environment using the principle of *stigmergy*[2]." The BaBe agents are shown in Figure 12 below.

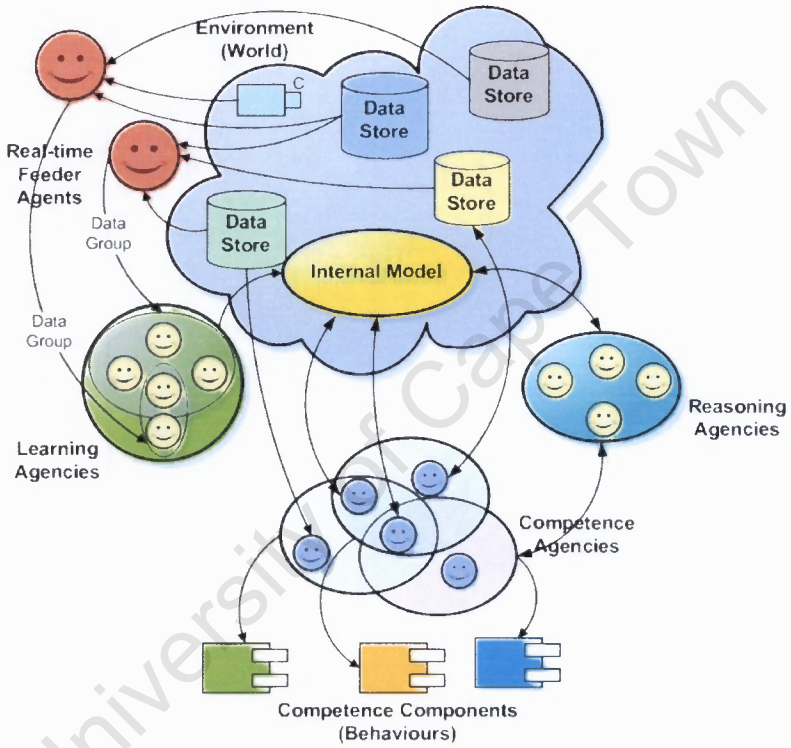


Figure 12: The BaBe Adaptive Agents

[1] The phenomenon of emergence means that the behaviour of a system can not be explained by observing the behaviour of the individual components that make up the system. In addition, emergence is generally an unexpected behaviour.

[2] *Stigmergy* is a method of communication in complex adaptive systems in which the individual parts of the system communicate with one another by modifying their local environment.

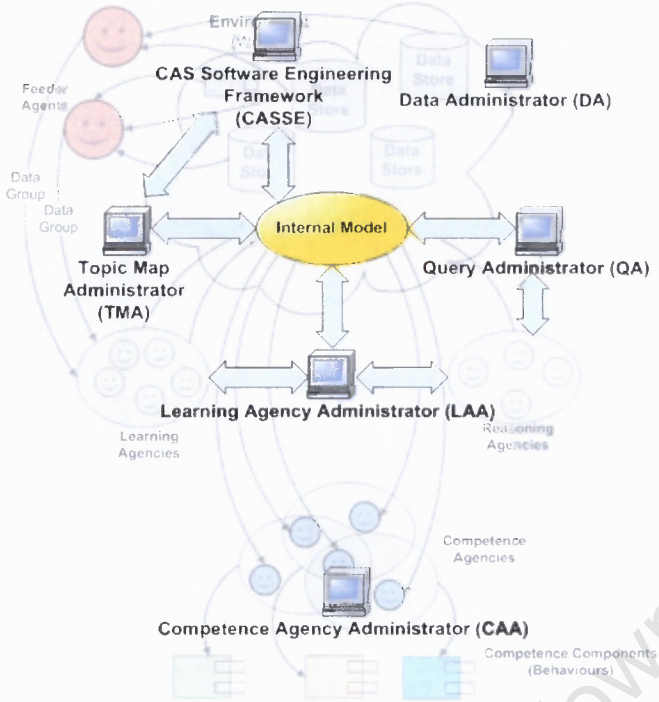


Figure 13: The BaBe Front Ends

Figure 13 above shows the BaBe front ends that are available to the end user. The BaBe Framework is discussed in detail by Potgieter in [62].

2.2.7 XMLBIF

XMLBIF is an Extensible Mark-up Language (XML) based Bayesian Network Interchange Format [55]. The format was primarily designed by Fabio Cozman, with contributions from Marek Druzdzel and Daniel Garcia, in an attempt to establish a common format for the exchange of knowledge and experimental results in terms of BNs [55]. Previous attempts to establish a common framework, e.g. BIF and BNIF, were not very successful.

The basic structure (DOCTYPE) of the XMLBIF format is shown below:

```
<?xml version="1.0"?>
<!--DTD for the BIF format-->
<!DOCTYPE BIF [
        <!ELEMENT BIF ( NETWORK )*>
        <!ELEMENT PROPERTY (#PCDATA)>
        <!ELEMENT TYPE (#PCDATA)>
        <!ELEMENT VALUE (#PCDATA)>
        <!ELEMENT NAME (#PCDATA)>
        <!ELEMENT NETWORK
          ( NAME, ( PROPERTY | VARIABLE | PROBABILITY )* )>
```

```
        <!ELEMENT VARIABLE ( NAME, TYPE, ( VALUE |  PROPERTY )* ) >
        <!ELEMENT PROBABILITY
          ( FOR | GIVEN | TABLE | ENTRY | DEFAULT | PROPERTY )* >
        <!ELEMENT FOR (#PCDATA)>
        <!ELEMENT GIVEN (#PCDATA)>
        <!ELEMENT TABLE (#PCDATA)>
        <!ELEMENT DEFAULT (TABLE)>
        <!ELEMENT ENTRY ( VALUE* , TABLE )>
]>
```

An iteration of an actual XMLBIF file that was used during this research can be found in Appendix E. An example of a BN specified using XMLBIF is shown below in Figures 14 and on the next page in Figure 15:

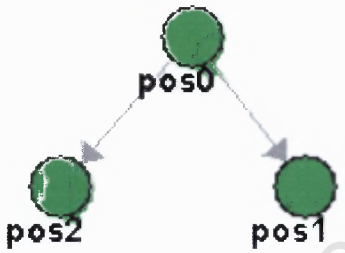


Figure 14: The structure of the example BN using JavaBayes [57]

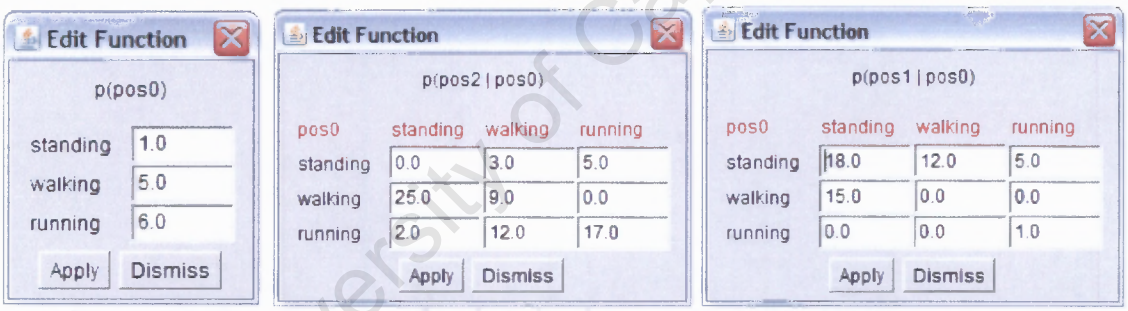


Figure 15: The parameters of each node of the example BN using JavaBayes [57]

The XMLBIF is shown below:

```
<BIF VERSION="0.3">
 <NETWORK>
  <NAME>block0Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos0</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
```

```
  <VARIABLE>
    <NAME>pos1</NAME>
    <OUTCOME>standing</OUTCOME>
    <OUTCOME>walking</OUTCOME>
    <OUTCOME>running</OUTCOME>
    <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
    <NAME>pos2</NAME>
    <OUTCOME>standing</OUTCOME>
    <OUTCOME>walking</OUTCOME>
    <OUTCOME>running</OUTCOME>
    <PROPERTY>position = (10, 0)</PROPERTY>
  </VARIABLE>
  <!--Probability distributions-->
  <DEFINITION>
    <FOR>pos0</FOR>
    <TABLE>1 5 6</TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>pos1</FOR>
    <GIVEN>pos0</GIVEN>
    <TABLE>18 15 0 12 0 0 5 0 1</TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>pos2</FOR>
    <GIVEN>pos0</GIVEN>
    <TABLE>0 25 2 3 9 12 5 0 17</TABLE>
  </DEFINITION>
 </NETWORK>
</BIF>
```

Some BN modelling tools, e.g. JavaBayes [57], have already integrated XMLBIF into their implementations, while other modelling tools, e.g. Hugin, show promising signs of adopting the format in the near future.

2.3 Conclusion

This Chapter presented some of the state of the art techniques used to develop a video surveillance system. In addition, it highlighted the fact that most video surveillance systems have an IU component that is used to model a chosen environment and extract data out of the environment and an AI component that acts on that data to perform certain surveillance tasks.

The first part of the Chapter focused on the techniques used to develop the IU component of a surveillance system which are related to the research conducted during this dissertation. The areas that were covered are: developing a BM, extracting correct foreground pixels, segmenting those pixels to form homogenous regions or blobs to represent the individuals that were in the environment, classifying those blobs, tracking those blobs and understanding the actions that the blobs were performing. In addition, a discussion about each area was presented where the advantages, disadvantages and points of note for each method were elaborated on. Furthermore, a few evaluation techniques were discussed which can be applied to each part of an IU component in order to evaluate its performance, robustness and accuracy.

The second part of the Chapter focused on the techniques that can be used to develop the AI component of a surveillance system which are also related to the research conducted during this dissertation. In particular the areas that were covered are: probability theory, modelling and knowledge representation, probabilistic reasoning and BNs. In addition a short discussion was presented about BaBe.

Chapter 3

Related Research (Literature Review)

Video surveillance systems generally combine the research areas of IU and AI in order to create a model of a chosen environment and to perform surveillance tasks on that environment model. Various approaches have been used by researchers to combine IU and AI techniques to form such video surveillance systems. These surveillance systems are continually improving in their accuracy, efficiency and robustness as new advances in both IU and AI are discovered. In addition, recent efforts to evaluate such surveillance systems have also helped in their overall performance improvement.

This Chapter presents some of the work conducted by other researchers in the field of video surveillance that is related to the research that was conducted during this dissertation and that combines the IU and AI techniques that were presented in Chapter 2. The first part of the Chapter focuses on a few state of the art video surveillance systems and their implementation, as well as the general trend that video surveillance has followed recently. The second part of the Chapter focuses on the varied use of Bayesian Networks in several video surveillance systems.

3.1 Video Surveillance Systems

Over the years several researchers have attempted to tackle the complex problems involved in developing a video surveillance system with varying results. Some of the surveillance systems and solution to the problems are presented below:

3.1.1 Background Subtraction and Object Segmentation

Although a few researchers, e.g. Lipton *et al.* [7], still use the classic temporal differencing background subtraction method, i.e. using consecutive frames to extract motion regions, most researchers have adapted background subtraction methods based on the work of Wren *et al.* [15]. Pfinder [15] revolutionized background subtraction, BM formation and object segmentation by using Gaussian distributions to model the environment and the objects in the environment. The previous problems experienced by non-adaptive background subtraction, e.g. appearance of ghosts and illumination variation, became a thing of the past. However,

their implementation was designed for an indoor environment. When used in an outdoor scene, their implementation still had problems with moving backgrounds, dynamic backgrounds and vast illumination changes.

Many researchers adapted and modified this revolutionary technique in order to suit their environmental needs. Friedman and Russell [1] extended the method by using an unsupervised incremental EM method to learn the mixture of Gaussian model in order to segment cars on a road. In addition, it appeared that Friedman and Russell [1] were able to handle the problematic effects of shadows by having a separate pixel class for shadows so that the shadows could be detected. Stauffer and Grimson [4] adapted the method so that multi-modal backgrounds could be modelled by using a mixture of Gaussians to monitor an outdoor environment. This implementation minimizes the effect of background clutter and drastic illumination variation. In addition it accommodated dynamic backgrounds because objects that were briefly introduced into or removed from the environment would not corrupt the BM.

3.1.2 Surveillance Systems

Several surveillance systems have been developed over the years with continuous enhancements being added to the systems. Existing surveillance systems range in the type of objects that they track and the type of activity that they monitor. Bodor *et al.* [2] track individuals in an outdoor environment and recognize several simple actions, e.g. walking, running, loitering and lying down.

Similarly, Stauffer and Grimson [4] developed a surveillance system that monitors an outdoor environment. In addition to monitoring individuals, their system can monitor groups of people and cars. However, their system is based on the use of motion tracks which limits the type of activities that they can recognize. Lou *et al.* [11] also use motion tracks in their surveillance system. Sidenbladh [6] uses motion information in the form of optical flow in the implemented surveillance system that can differentiate between people and cars.

Several other surveillance systems were designed to monitor various objects in the environment. Ivanov *et al.* [10] monitor people and cars and the interactions between them. In addition, their system detects several complex actions that occur during the interactions.

Similarly, the video surveillance system developed by Park and Trivedi [19] detects interactions between people and vehicles. Part *et al.* [18] and Chen *et al.* [21] also monitor interactions between objects, but their focus is on multiple person interactions.

Unusual activity recognition has come to the forefront of the researching community in recent years. This is mainly because focus has been shifted to the security of populated areas due to the fear of terrorist attacks. Stauffer and Grimson [4] can detect unusual motion tracks by comparing new motion tracks to the common motion tracks that were learnt by their system. In addition, [43, 44, 45 and 46] were designed to detect abandoned luggage in crowded areas. The system developed in [14] detects unusual activity on an airport apron which is also a high risk area in determining airplane sabotage and hijackings.

In addition, monitoring of mass transit areas and the actions occurring at the mass transit areas has recently emerged through European Union sponsored projects, e.g. monitoring of airports by AVITRACK [14], and the workshops conducted by PETS, e.g. [43, 44, 45 and 46]. In addition one of the focuses of the PETS [41] workshop in 2006 was the detection of abandoned luggage at crowded areas. Similarly, [16] detects the actions that occur in front of a shop in a shopping complex.

Monitoring of restricted areas has also been implemented by some researchers with the classing set-up of the system highlighted by Bodor *et al.* [2] who use proximity information and prior knowledge of the location of the restricted areas to integrate them into their system for monitoring. Alarms are typically raised if entry into the restricted areas is detected.

The use of multiple sensors in a video surveillance system can have many benefits especially if the sensors are complementary and cover each other's negative points. A good example of this would be to use a sensor that can measure environment depth in collaboration with an ordinary camera in order to detect camouflage in the scene. The use of multiple sensors is evident in the work of certain surveillance systems, such as [4], [14] and [52]. In addition Seibert *et al.* [39] use multiple sensors to monitor a sea port by monitoring tracks of movement made by the ships moving through the port.

Finally, some surveillance systems are focused on the task of facial recognition, e.g. Wei *et al.* [3] and the extraction of facial features [17]. Other systems focused on aiding human

operators by changing the region of interest (ROI) into a representation that is more likely to catch a human operator's attention, e.g. the work of Wang *et al.* [9].

3.2 PETS

The IEEE holds an annual international workshop on Performance Evaluation of Tracking and Surveillance (PETS) [41] systems. Although the name suggests that evaluation is the main task of the workshop, several researchers are attracted by the workshops to demonstrate the state of the art techniques in surveillance systems. The following workshops have been held so far:

- 2000 Outdoor people and vehicle tracking (single camera)
- 2001 Outdoor people and vehicle tracking
- 2002 People tracking (and counting)
- 2003 Outdoor people tracking
- 2003 Annotation of a smart meeting. Includes facial expressions, gaze and gesture/action
- 2004 CAVIAR people scenarios
- 2005 Challenging detection/tracking scenes on water
- 2006 Surveillance of public spaces, detection of left luggage events

In addition to holding the workshops, PETS has provided sample data sets and evaluation metrics that can be used by researchers to compare and evaluate their video surveillance systems. There is also an online evaluation facility were researchers' evaluation results are submitted and reviewed by PETS.

3.3 AFORGE

AFORGE, or AForge.NET, is an open source framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence [56]. The main designer of the framework is Andrew Kirillov. In addition, the framework was developed in C# and provides methods and functionality in the following areas:

- image processing
- neural networks
- genetic algorithms
- machine learning

3.4 Bayesian Networks in Video Surveillance Systems

Bayesian Networks (BN)s have recently attracted interest by the video surveillance community for their powerful statistical representation and data modelling capabilities. As a result, more researchers have adapted the use BNs in their video surveillance applications. Some of these implementations are highlighted below.

3.4.1 Facial Expression Recognition

Cohen *et al.* [17] highlight the fact that the most expressive way humans display emotions is through facial expressions by building a facial expression recognition system using a semi-supervised BN. A vector of motion features of certain regions of the face is comprised and the features are used as the input to a BN. A snap-shot of their system is shown in Figure 16, below:

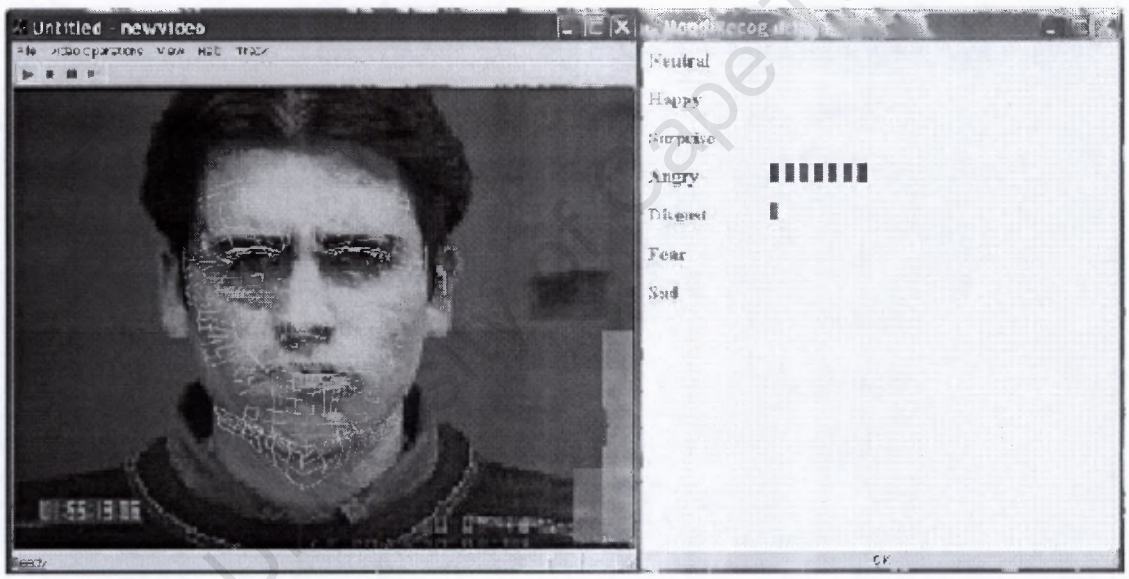


Figure 16: A snap shot of the facial expression recognition system developed by [17]

3.4.2 Recognizing Interactions

Park and Aggarwal [18] use a hierarchy of BNs for recognizing two person interactions. The people in the environment are segmented into their body parts, i.e. the head, hand, leg and torso, and a separate BN is used for estimating the pose of each body part. These separate BNs are integrated into a hierarchy of BNs in order to estimate the overall body pose of the individual people in each frame. The interactions are recognized by incorporating the overall

body pose estimation and spatial/temporal constraints about relative positions and causal relations between the people involved. The structure of the BNs was manually constructed because of insufficient training data.

Chen *et al.* [21] use a BN to create a system that can automatically extract and classify important antecedents of psychosocial and health outcomes of elderly people by detecting the frequency, duration and type of social interactions of the patients with one another and their caregivers in a nursing home. An Ontology[3] of social interactions is specified and mapped onto the BN in order to recognize the interactions.

3.4.3 Object Tracking

Some researchers have used BNs for tracking objects and for reasoning about tracking data that is gathered. Kang and Cho [51] use a BN to implement an adaptive object tracking method by using the BN to combine cues from colour likelihood and edge likelihood which are used for object tracking. Petrushin *et al.* [52] use a Bayesian framework for the integration of evidence from multiple sensor sources in order to localize and track people in the environment.

3.4.4 Human Action Understanding

Baker *et al.* [12] use a Bayesian framework to explain how people think and how they use observed behaviours to predict actions. In their implementation, the behaviour of people is explained by taking the intentional stance, i.e. people are treated as rational agents whose behaviour is causally governed by beliefs, desires or other mental states that refer to objects, events, or states of the world. Furthermore, they assume that the rational agents aim to perform their desires as optimally as possible by constructing plans or sequence of intended actions based on their beliefs or knowledge about the environment.

[3]An Ontology is a vocabulary that represents the objects that are present in an environment and the relationships amongst the objects. It is discussed in detail by Russell and Norvig [48].

3.4.5 Activity Recognition

Ribeiro and Victor [23] use a BN to detect short term human activities that are happening in the environment. The BN is used to model the activities that can happen in the environment. Gaussian mixtures are used to model the likelihood function of each action in the BN. In addition, the EM algorithm is used to estimate each likelihood function. Similarly, a BN is also used by Lv [46] for event modelling and recognition.

3.5 Conclusion

This Chapter presented the work of several researchers in the field of video surveillance that is related to the research carried out during this dissertation. In particular the Chapter presented the way the IU and AI techniques that were presented in Chapter 2 have been combined to develop state of the art surveillance systems. In addition, the Chapter presented the trend that current video surveillance systems are following. Furthermore, the use of BNs in such surveillance systems has also been presented.

The next Chapter presents the prototype video surveillance system that was developed to demonstrate the research that was conducted during this dissertation.

Chapter 4

Research Methodology

This Chapter clarifies the focus of this research and presents the methodology that was used to achieve the research goals that were set out. In addition, it discusses the way the IU and IA techniques that were presented in Chapter 2 were combined to build the prototype surveillance system. Hence, the design and implementation of the developed prototype surveillance system are presented, as well as an in-depth explanation and discussion about the prototype system components.

Furthermore, a discussion about how the monitored environment was selected, how anomalies were specified and how the chosen human actions were selected is also presented.

4.1 Research Focus and Overview

The focus of this research was to detect and predict anomalous behaviour of individuals in a specific environment and to detect and predict entry into restricted areas. Hence, one can clearly see that the focal point of the research was the application of AI techniques to achieve the desired goals of the research. However, in order to perform such a task a suitable environment had to be chosen and modelled. Furthermore, extraction of vital information about the environment, such as the people in the environment and their actions, had to be performed.

A video camera was chosen to capture the environment as a sequence of images. This decision inevitably required an IU component that could extract information about the environment from those images. Thus it can be stated that the IU component was not the focal point of the research but instead it can be viewed as a capacitating component that provided data capturing and information gathering capabilities for the research purposes. In light of that fact, the IU component was built as a data gathering component specifically for the purpose of this research. In addition, the IU component was built in a manner that would make replacing it with a more sophisticated IU component as easy as possible because there are constantly new IU techniques that are developed which are more robust and more

efficient than the current ones. Hence it would make sense to use these techniques in an IU system.

The research as a whole had several important steps that had to be carried out. These are listed below:

1. Requirements gathering and decision making
    a. Choosing an environment where the research was to take place
    b. Deciding on the way to model the chosen environment
    c. Deciding on the number of people that would be monitored
    d. Deciding on what kind of actions to recognize and keep statistics about
    e. Deciding on how to specify the anomalies that would be detected and predicted
    f. Deciding on how to specify restricted areas.
2. Implementation of the prototype surveillance system
3. Testing and Evaluating the detection and prediction of actions done by the prototype surveillance system

These steps were performed by following a refined waterfall research model, which is shown in Figure 17 below:

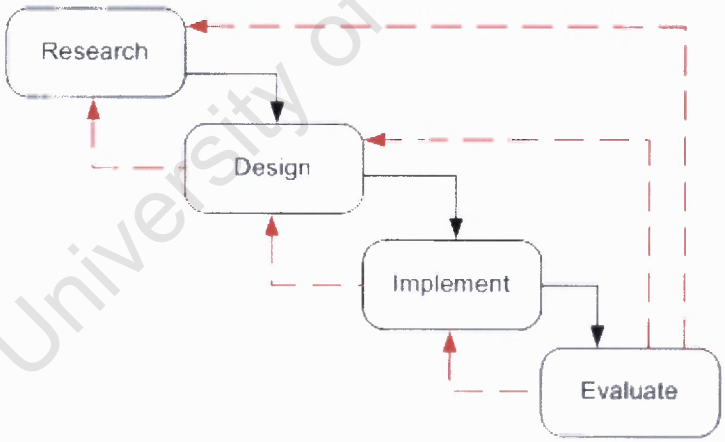


Figure 17: The refined waterfall research model used during this dissertation

Each requirement and research goal was initially well researched so that an overall view of the requirement could be established. In addition, the research was conducted so that the most suitable techniques to accomplish the requirement were well understood. Once each

requirement was well researched the design decisions could be established and the implementation followed that. The last step was to evaluate the developed solution for each requirement. If any problems were encountered in any of the steps backward navigation was done to investigate a more refined or different approach to solving the requirement problem.

The following sections describe each aspect of the research that was conducted.

4.2 Camera Description

The camera that was used to capture information about the environment is an AXIS 210 camera [60]. The camera is an IP network camera that can easily be connected to a computer or an existing network. This proved to be a very useful feature because the camera could be connected to the Agents Research Lab network. In addition, the camera delivers images in Motion JPEG (MJPEG) or MPEG-4 format at a customizable frame rate. A picture of the Axis 210 camera is shown in Figure 18 below:

Figure 18: The Axis 210 camera

There are two main drawbacks that were discovered about the Axis 210 camera. The first is the fact that the camera has an inbuilt function that automatically adjusts the contrast of the images that are recorded. The effect of this feature can be minimized, but it cannot be switched off completely. The second drawback is the fact that the camera does not perform well in badly lit environments. In particular the frame rate and the image quality are compromised.

4.3 Programming Language

The programming language used to develop the prototype surveillance system was C#. The main reason that the language was chosen was that it combined aspects of Java and C++. For instance garbage collection is performed automatically which is similar to Java while the ability to use pointers is similar to C++. The use of pointers in particular gave a great performance advantage over Java when accessing and manipulating image data.

4.4 Data Format

The data used during this research comprised of images captured by the Axis video camera. The images were received in Motion JPEG (MJPEG) format which is a continuous flow of JPEG images. This meant that the images needed some pre-processing before they could be used. The MJPEG format that the camera uses to transmit its images is specified in the AXIS MEDIA CONTROL SDK document [60] and is shown below.

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=myboundary\r\n
\r\n
--myboundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15656\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: 14978\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
Content-Type: image/jpeg\r\n
Content-Length: 15136\r\n
\r\n
<JPEG image data>\r\n
--myboundary\r\n
.
.
.
```

Before the images could be processed to extract pixel data, all the extra information about the image that surrounded the pixel data had to be stripped away. This process required the extraction of the pixel data length, the Content-Length field, from the header and then

removing the header and footer of each image so that only the pixel data remained. At this stage only the pixel data remained for each image and processing of the pixel data could be performed by the IU component. In addition the *--myboundary* field specified the end of each image in the sequence.

4.5 Choosing an Environment to Monitor

Choosing an environment to model and monitor may not always be a trivial task as was discovered during this research. The first choice that had to be made was to decide between an indoor and an outdoor environment to monitor. Several factors were considered before a decision was made. These factors are shown in the table below.

Factor	Indoor Environment	Outdoor Environment
Illumination variation	Can be minimized in an indoor environment by having a constant light source	Can cause drastic variations when attempting to model an outdoor environment
Other environmental factors e.g. rain	Environmental factors would not affect an indoor environment	Environmental factors such as rain can make modelling an outdoor environment very challenging
Background clutter or movement	Minimal to no background clutter	There can be lots of background clutter especially if trees are in the background
Dynamic Backgrounds	There would be less addition and removal of objects because the environment can be controlled	It is much harder to control the addition and removal of objects in an outdoor environment
Effect of camera's automatic contrast adaptation	The camera would use more automatic contrast adaptation especially when a white background is used and a dark foreground object passes through it	The camera would use less automatic contrast adaptation because it performs better in environments where there is significant light
Shadows	Can be minimized by having a central light source located above the environment	Can cause errors when modelling by being incorrectly detected as foreground objects
Camouflage and holes in foreground objects	The objects tend to take up more space in the image and thus are more prone to camouflage problems resulting in holes or disjointed foreground regions	The objects tend to take up much less space in the image, hence camouflage effects are less

Computational Complexity of algorithms	Overall the computational complexity of the IU component would be less because less factors need to be accounted for and would be easier to implement in a real time manner	Overall the computational complexity of the IU component would be greater because of all the factors that need to be accounted for and would need a really fast computer to perform operations in real time

Table 1: The comparison of indoor and outdoor environments for the purpose of this research

After considering all the factors presented above, a decision was made to monitor an indoor environment. The reason for doing this was to minimize the complexity of the IU component. In addition, an indoor environment would ensure that the camera would not come under security threat. This decision meant that factors such as background clutter, rain and dynamic backgrounds could be controlled and thus their impact on the environment would be minimized. In addition, the effects of shadows would be minimized and an attempt at monitoring an environment in near real time could be made because less resources and computational complexity would be needed for monitoring an indoor environments. However, the decision also meant that the automatic contrast adaptation by the camera would have to be dealt with as well as the effects of camouflage.

The next step was to experiment with the chosen environment in order to get a base idea about the way the environment reacts. Initially monitoring was done during daylight hours, but after some time it was evident that illumination variation caused by clouds was too much for a simple IU component to handle and a decision was made to monitor an indoor environment at night time with a central light source. In addition, this decision would further minimize the effects of shadows. The Agents Research Lab proved to be just the environment that was needed. It had sufficient space for monitoring peoples' actions and it was well lit with central light sources.

Such a decision was justified because it eliminated the complex algorithms needed to handle multi-modal backgrounds such as the ones caused by illumination variation. Furthermore, the decision would allow the IU component to be built as a data capturing component and would allow enough time for the research to focus on the AI techniques. The chosen monitoring environment is shown in Figure 19 on the next page.

Figure 19: The Agents Research Lab

4.6 Prototype Overview

This section presents an overview of the prototype video surveillance system that was developed during this research. A high-level conceptual view of the prototype video surveillance system is shown in Figure 20 below:

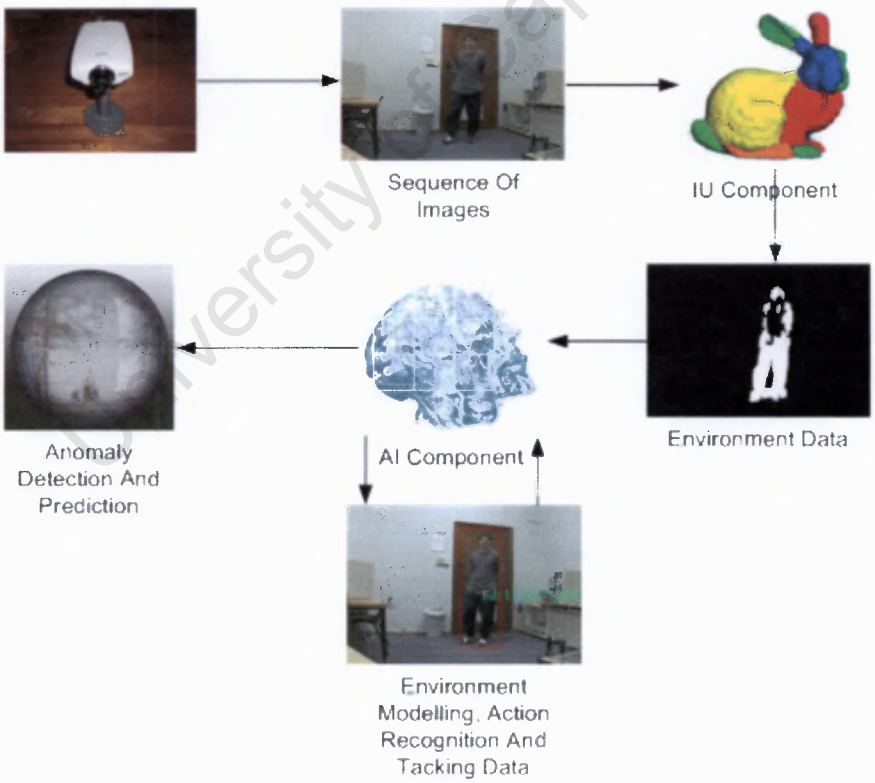


Figure 20: A high-level component view of the prototype surveillance system

The prototype surveillance system was developed so that people in an environment could be tracked and monitored. The actions that the people performed were analysed in order to detect and predict anomalous behaviour as well as entry into restricted areas. The aim of the IU component was to provide learning data for the AI component by segmenting foreground pixels into blobs in each frame. Bounding boxes were formed around those blobs so that they could be used by the AI component to determine the position of each blob. From that information the action that was being performed was recognized. Predictions were based on learnt behaviour patterns of the blobs in the environment. In addition, the prototype system was developed to normally monitor a single person at a time in the environment, although multiple people monitoring was possible as long as there was no overlap or occlusion of the people. The following sections describe each component in detail. A more detailed class diagram of the prototype video surveillance system can be found in Appendix A.

4.7 Image Understanding Component

This section discusses the design decisions that were made about the IU component and also presents the implementation of the IU component of the prototype surveillance system.

4.7.1 Design Decisions

This section presents the design decisions that had to be made in order to implement the IU component of the prototype surveillance system.

Choice of Background Model

Initially, choosing the BM appeared to be a trivial task because the IU component was only supposed to provide learning data for the AI component and not be too sophisticated. Hence, a simple temporal differencing method, such as the one used by Lipton *et al.* [7], was developed to extract foreground pixels. After a brief pilot trial period it was discovered that only parts of the blobs in the scene were being detected when the blob was moving and blobs were not being detected at all when they were stationary. In addition, ghosts appeared in the difference image which was not a desirable feature. These are classic problems with temporal differencing techniques.

Hence, a static averaged BM was the next solution. This too proved to be unsuccessful because erroneous foreground pixels began to creep in over time due to the fact the BM was not adapting to the environment and was not being re-initialized. This lead to a thorough analysis of the way the environment behaved in order to establish the most appropriate BM and background subtraction algorithm.

The following facts about the environment were established:

- Contrast adaptation by the camera – this caused a lot of erroneous foreground pixels to be detected and had to be dealt with accordingly
- Minimal variation in illumination was experienced because monitoring was performed at night. In addition the multi-modal background was reduced to a singular background.
- There was no background clutter
- Dynamic backgrounds could be controlled.

From analysing these facts the first hypothesis was to use an averaged BM which is re-initialized at certain time periods and to implement a threshold to handle the illumination variation. However, the contrast adaptation by the camera dismissed this theory and a more sophisticated BM had to be implemented. The common method of modelling pixels using Gaussians was considered, but a decision had to be made regarding the use of a single Gaussian or a mixture or Gaussians. The facts that the background was not a multi-modal background and the fact that a mixture of Gaussian implementation was more computationally expensive as compared to a single Gaussian, led to the single Gaussian method being chosen.

A Gaussian BM proved to be able to handle the contract adaptation of the camera in a satisfactory manner and was chosen as the BM. The implementation of the Gaussian BM is detailed in the next section.

Shadows

In the early stages of development, shadows proved to be a major problem, however after switching to monitoring at night the effects of shadows were greatly reduced. Nevertheless minor shadows were still evident and had to be dealt with accordingly. A modified version of the YUV analysis approach used by Pfinder [15] for shadow removal was implemented and is presented in the implementation section.

Number of People to Monitor

A decision was made to monitor only individuals in the environment instead of groups of people because the IU component was not very sophisticated and was not envisioned to monitor groups. In addition, handling occlusion of individuals was not envisioned as a functionality of the IU component because it was assumed that normally only one individual would be in the environment at a time. Hence if several individuals were in the environment at the same time and were close to each other the surveillance system would not be able to tell them apart and would simply merge them into one big blob. Occlusion handling, as well as monitoring group activity or dynamics, can be tasks for future work.

4.7.2 Implementation

This section discusses the implementation of the IU component in some detail.

Image Understanding Component Overview

The IU component had 5 major parts that enabled it to act as a data capturing and gathering component. A high-level view of the way the parts function is shown in Figure 21 below:

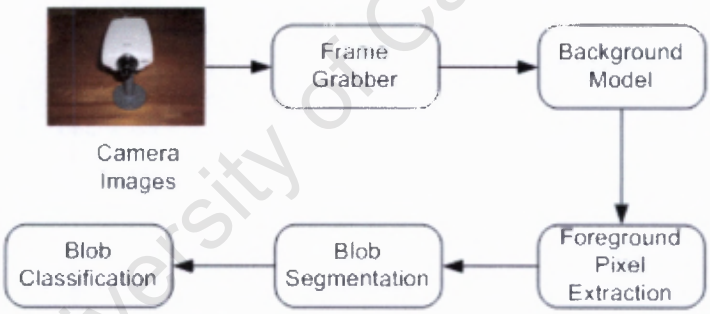


Figure 21: A high-level view of the IU component

Each part is discussed in the following sections.

Frame Grabber

A frame grabber was developed in order to extract individual images from the MJPEG image sequence that the Axis 210 camera was streaming and also to format the image data into usable data matrices. The image resolution that was used was 352 x 240 pixels and the frequency of the images was 7 frames per second. Once the images were formatted into data matrices they were fed into the BM part for initial processing.

Background Model

Following the investigation about how the chosen environment behaved, a decision was made to use a single Gaussian BM, such as the one used in Pfinder [15], to represent the environment. In order to accomplish, this several steps had to be performed. These are listed below.

1. Initialization of BM

An initialization period is commonly used in literature to establish an initial BM which is then adapted to represent the changes in the environment. Some of the researchers that have used initialization periods are [2], [4], [14] and [15]. The RGB colour space was used to represent the value of each pixel. In addition, an assumption was made that each colour channel was independent so that a simplified co-variance matrix, consisting of a diagonal of each colour channels variance value, could be used in the Gaussian BM calculations. Although this is certainly not the case in reality, a trade-off between accuracy and computational time was achieved. Several researchers, including [4] and [14], have used such an assumption and state that the difference between the approximate values derived from such a co-variance implementation and a proper co-variance matrix is minimal and insignificant.

During the initialization period, statistics were gathered about the mean value (μ) and variance ($\sigma^{2)}$ of each pixel's individual colour channels. These parameters were then used to represent the Gaussian distribution of each pixel. Each pixel's mean values were then combined into an image so that a representation of the environment background could be established and used for comparison with new frames that were received from the camera. In addition, each pixel's variance values were stored in a vector so that they could be easily accessed when needed.

2. Updating the BM

Updating of the BM was done in accordance with common practice in literature by using an exponential equation, such as Equation 2.5 which was used by [4], [14] and [15]. Several points had to be considered about the way the BM would be updated. Firstly, including foreground pixels into the BM would corrupt the BM, so foreground pixels were not used in the update process.

After a few pilot tests it was discovered that the mean and variance of each pixel varied in different ways. In addition it was discovered that outlying values (at the limit of 2.5 standard deviations) were shifting the mean value significantly more than the variance value of each pixel. At the same time it was discovered that when the difference between a new frame and the BM was less than 1 standard deviation both the mean and variance values of each pixel were shifting towards 0.

Thus a decision was made to control the way the mean and variance values were being updated. The mean value was updated when the new pixel values were between 0.8 and 1.5 standard deviations of the BM mean while the variance values were updated when the difference between the BM mean and the new pixel values was between 1 and 2.25 standard deviations. In addition, all difference values between the BM model and the new pixel that equalled 0 were set to equal 0.1. In such a way the shifting effect of outlier values was minimized.

Foreground Pixel Extraction

Once the BM was established and was reasonably stable the extraction of correct foreground pixels could be done. The following steps were taken to make sure that the correct foreground pixels were extracted.

1. Extraction of a Difference Image

Gaussian BM Subtraction was used to detect all foreground pixels. All the new pixel values that were within 2.5 standard deviations of the BN model were considered to fall within the background range. Pixel values that did not fall within that range were considered to be foreground pixels. The standard deviation value was calculated by converting each pixel's Gaussian distribution into a standard normal variable using Equation 2.8. However, a slight modification of the equation was used. Instead of simple subtraction the Mahalanobis distance [27] between the pixel values was used because each pixel's mean value was three-dimensional, i.e. the red, green and blue component. This operation resulted in a difference image containing all the pixel values that were more than 2.5 standard deviations away from the BM model.

2. Formatting the Difference Image

The difference image needed to be converted into a binary image containing only black (background) and white (foreground) values so that the rest of the IU parts could process it efficiently. This was accomplished by applying a threshold operation, such as the one discussed by [25] and [47], on the difference image. The difference image was processed in a pixel-by-pixel manner to form the binary difference image. The pixels that had a value greater than 0 for any of its colour channels were assigned a white colour (255) whilst the other pixels were assigned a black colour (0).

3. Noise Removal

The binary difference image revealed the presence of random salt-and-pepper noise. Initially, a median filter, such as the one described by [25] and [47], was used on the new frame before Gaussian BM subtraction to try and minimize the effect of the noise. However, the application of the median filter was time consuming and made the whole IU slow and less efficient. As a result a noise removal process using morphological operators, presented in [25] and [47], with a box shaped structuring element, that represented an 8-connected pixel, was used to remove the salt and pepper noise that was visible in the binary difference image. Specifically the opening operation was used to remove small noise regions in the image and random salt and pepper noise. This noise removal technique proved advantageous in the blob segmentation process as well.

Blob Segmentation

Segmentation of blobs required two crucial steps. The first was to identify the foreground pixels that formed homogenous regions or blobs and the second was the removal of shadows regions.

1. Segmenting Homogenous Foreground Pixels into Blobs

Once the erroneous foreground pixels were removed segmentation of the regions in the image into blobs could be attempted. Initially a single pass connected component algorithm provided by Aforge [56] was used to segment the foreground pixels into regions or blobs. However, after a few pilot tests it was discovered that a few blobs had holes in them which

were caused by camouflage with the background, whilst other blobs had been separated into a collection of smaller parts that were in close proximity.

The first problem of holes appearing in some blobs was solved by applying a closing morphology operation [25 and 47] to close up the holes. The second problem of closely located collections of small blobs which were part of a single blob required two steps to solve. The first step was to merge closely located small blobs into a larger blob. A bounding box was worked out for each small blob and was used to represent the boundary of the blob.

The Euclidean Distance [61] was worked out between closely positioned small blobs to determine which ones were close enough to merge. 8 points along the boundary of each blob's bounding box were considered in the Euclidean distance calculation, with the minimum Euclidean distance of the 8 points being used to represent the distance between the small blobs. If this Euclidean distance was smaller than a threshold value then the small blobs were merged.

The threshold value (20 pixels) was determined by evaluation of the small blobs' proximity and is discussed in Chapter 5. This process is shown in Figure 22 below:

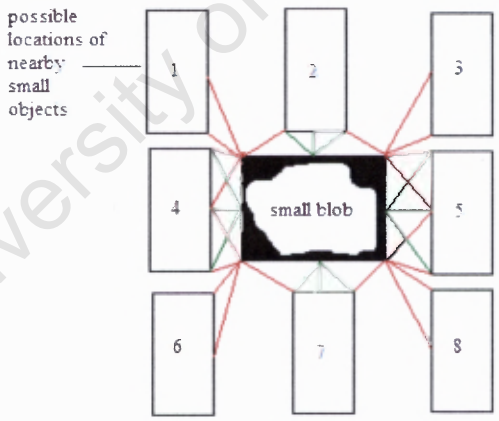


Figure 22: The methodology used to determine blobs in close proximity to each other

Such a method was justified because of the assumption that normally only individuals were monitored in the scene and when more than one individual was present in the environment the distance between the people was kept quite large. Thus merging of incorrect small blobs

was very rare. In addition, an assumption was made that each small blob's centroid would fall within the constraints of the small blob. Thus, initially the actual merging of the small blobs was done by drawing a line between the centres of the blobs.

However, after a few pilot tests it was discovered that the centroid of each small blob was not necessary within the foreground pixels of the small blob. As a result merging was not being done correctly. This situation is shown in Figure 23 below:

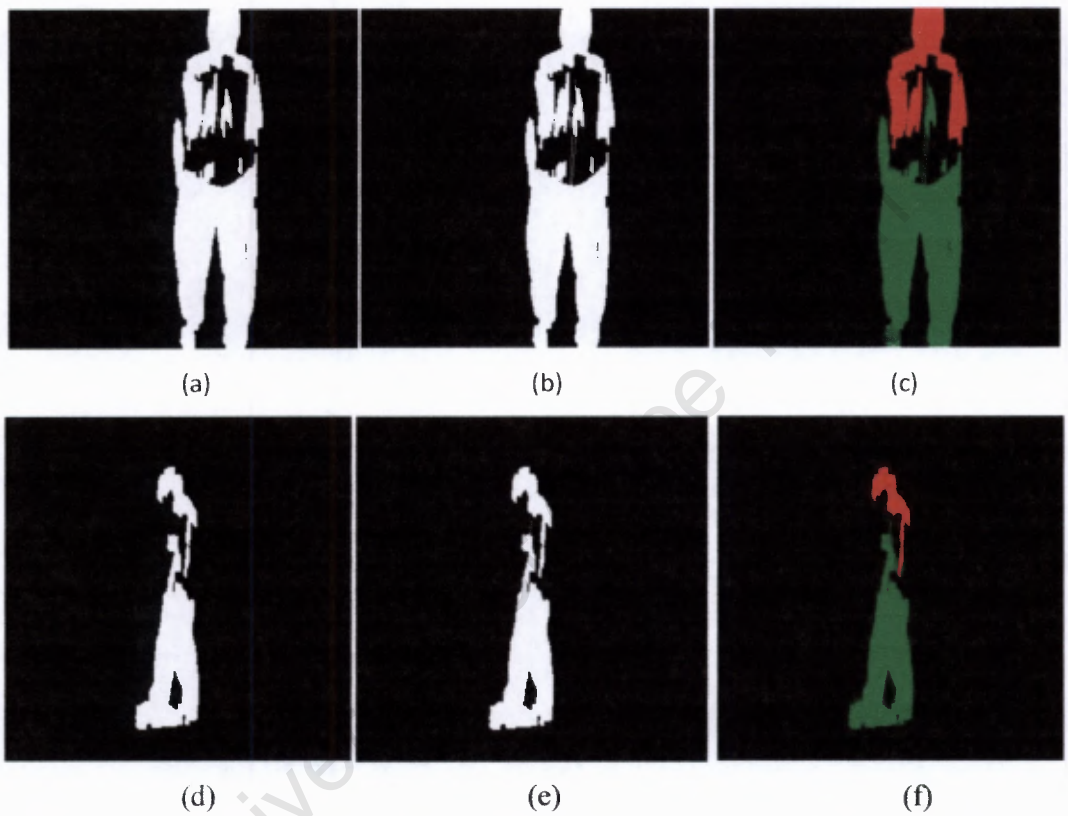


Figure 23: Two example sequences which demonstrate incorrect blob segmentation

The first picture (a and d) is before merging the small objects, the second (b and e) is after merging and the last picture (c and f) is the connected components that are detected. One can clearly see that the final segmentation results in two differently coloured regions instead of a single region. The reason for this is that the lines that are drawn between the two separate blob parts in images b and e do not connect the two blob parts. Hence, a decision was made to replace each small blob by a filled rectangle with the same dimensions of the small blob's bounding box. This was done to ensure that the small blobs would be correctly linked when a line was drawn in between them.

Once the small blobs were merged into larger blobs a second pass of the connected components algorithm was performed to correctly segment the larger blobs and get a more accurate representation of the individuals in the scene. This procedure seemed to segment the individual in the environment into the correct blob representation. An example of this procedure on an actual blob in the environment is shown in Figure 24 below:

Figure 24: The use of filled rectangles to represent blobs

2. Shadow Removal

Once the blobs were correctly segmented the appearance of small shadows in the environment became evident. This undesired problem was solved in a manner that is similar to the method used by Pfinder [15]. When a shadow covers a foreground pixel generally the brightness of the foreground pixel tends to become darker. This assumption intuitively meant that the RGB colour format that the BM and the new frame were in had to be converted into a colour format that had a representation of the brightness of the pixels. The YUV colour format was chosen because it represented the brightness value of a pixel. The following formula, adopted from [25], was used to convert RGB to YUV:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.1}$$

Thus, all the foreground pixel values that appeared darker than normal were processed in order to determine whether only the brightness value had changed. If this was the case then the pixel was assumed to be a shadow and was labelled as a background pixel so that its effect could be removed. Otherwise the pixel remained a foreground pixel. This approach reasonably detected and removed the small shadows that appeared in the environment.

Blob Classification

Once shadows were removed from the environment the classification of the blobs in the environment appeared to be a rather simple task. This was mainly because it was assumed that the only moving objects in the environment were people. Hence any blob that the IU component extracted was assumed to represent the people in the environment. However, after a few pilot tests to see if this assumption would hold it was discovered that noise regions, which were larger than the structuring element used in the noise removal morphology operation, were also being extracted by the IU component.

After a brief analysis of the problem, it was decided that the solution was to use the size of the blob that represented the person in the environment as a filter to remove any larger noise regions. In order to do that a bounding box that represented the size of each blob had to be formulated and used for comparison purposes. A modified version of the Aforge [56] method to get connected components was used to create the bounding boxes which surrounded the each blob. This method was similar to the one used to formulate bounding boxes when merging of closely positioned small blobs was done.

The next step was to perform evaluation tests to find out the approximate size of the blob that represented the person in the environment and the larger noise regions. The results for these tests can be found in Chapter 5. Briefly, it was discovered that the blob (including the small blob parts that were merged) that represented the person in the environment was larger in size than a 20 x 20 pixel rectangle whilst the size of the noise regions was smaller than that. Hence a threshold or filter was used to clear out any blobs that were smaller than a 20 x 20 pixel rectangle.

After this point the data about the blobs in the environment were passed onto the AI component for further processing.

4.8 Artificial Intelligence Component

This section presents the design and implementation of the AI component of the prototype surveillance system.

4.8.1 Design Decisions

This section presents some of the design decisions that were made during the implementation of the prototype surveillance system.

Dividing the Environment into a Grid

Ideally each position in an environment should be monitored, however this is not always possible because of the massive amount of data and processing power that would be needed to represent such a model. Often researchers use assumptions and simplifications to minimize the amount of data and processing power that is needed to model the environment.

Hence a decision was made to divide the environment into a grid of blocks in order to reduce the amount of data and processing that would be needed to model the environment but still keep the proximity relations of the environment. Such an approach is common in environmental and geographical studies and is highlighted by the fact that the earth is divided into a grid. Furthermore, the co-ordinates of any point on the earth are measured by a combination of longitude and latitude values. This idea is shown in Figure 25 below:

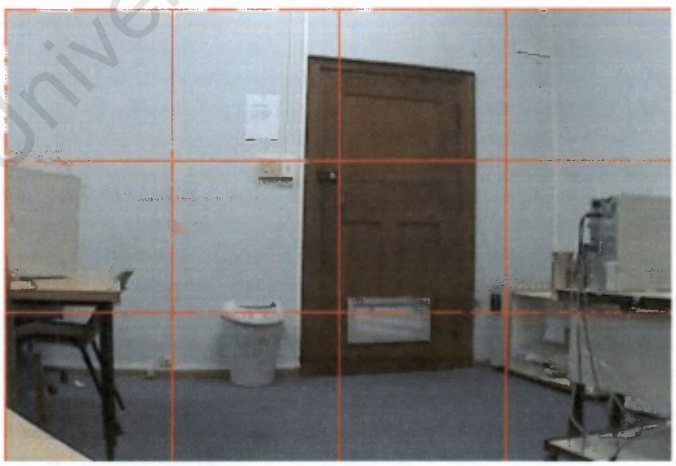


Figure 25: The environment divided into a grid of blocks

Each block can then be monitored in order to learn action patterns in that block and perform detection and prediction of anomalous behaviour in each block. In addition, by splitting the environment into a grid, the cause and effect relationships between actions in a specific block and the actions in the surrounding blocks can be modelled. Furthermore, splitting the environment into blocks makes representing restricted areas much easier. This idea solves a part of the second research question which is devoted to representation of environmental data so that it can be modelled by a BN without losing vital information about the environment.

Choosing the Actions to Monitor

There are several human actions and behaviours that have been monitored in the literature but not all of those could be monitored during this research. The main reason for this is that the surveillance system that was developed is a prototype system which is supposed to show a proof of concept and not exhaust all possibilities. Hence, a decision was made to model simple actions that could be extracted from tracking data alone with the view that complex action monitoring could be added in future work. In addition, the chosen actions had to have different characteristics so that recognizing the actions would not be a laborious and complicated task.

The environment was studied for a period of time to determine the most common simple actions that occurred with the following simple actions being identified:

- Walking
- Running
- Standing
- Sitting
- Entering the lab
- Leaving the lab

These simple actions were further assessed in order to choose the exact actions that would be monitored by the prototype system. Entering and leaving the lab can only be done through a single door; hence monitoring such actions would not give any valuable information about the way people behave in the environment as the actions are isolated at one point. Thus they could be discarded from the list of actions to monitor. Also, in this particular environment the

actions of standing and sitting could be viewed as being the same because they both imply that a person is stationary. Thus standing and sitting were combined into one action.

Hence the actions that were monitored by the prototype system were standing, walking and running. Although these actions are trivial simple actions, when combined with positional information they can describe the behaviour of people in the environment very well. Bodor *et al.* [2] demonstrated this concept in their surveillance system. Finally, the chosen actions had a common differentiating characteristic, namely motion, which would make the task of recognizing the different actions easier.

Specification of Anomalies

After choosing the actions that were monitored a decision had to be made about how anomalies would be specified. Hu *et al.* [5] suggest the use of a combination of probability reasoning, to detect behaviours with small probability as anomalous behaviours, and the use of prior rules to specify anomalous behaviour directly. Hu *et al.* [5] also suggest comparing behaviours to a set of learnt patterns of behaviours in order to detect anomalous behaviour.

These suggestions were combined in the prototype surveillance systems in order to specify anomalous behaviour. By observing the positions in the environment where the actions are taking place, certain trends become evident. Hence, the surveillance system can learn common trends and be able to identify actions that do not follow the common trends in each position in the environment. This point highlights the essence of the third research question.

Also, in some cases, certain actions should never happen in specific locations of the environment. This kind of information would be best represented by heuristic knowledge about the environment. In addition, detecting anomalous behaviours can aid any surveillance system. Thus the following anomaly specifications were used in the prototype surveillance system:

1. Learnt from Data

The ability to detect anomalies from learnt data is a very powerful tool and is shown in this prototype surveillance. The prototype system would learn the common trends in specific

locations in the environment and any action that does not fall into this trend would be classified as anomalous.

2. Heuristic Knowledge

A heuristic opinion was used to specify certain actions as being anomalous regardless of the learnt patterns by the system. This is a way of overriding the system in order to make sure that the specified actions are detected as being anomalous.

By specifying anomalous behaviour in these ways a framework for answering the third research question was formed.

Specification of Restricted Areas

The fact that the environment was split into blocks made the specification of restricted areas a very easy task. All the user had to do was to select the blocks which should be represented as restricted areas. The system would then monitor actual entry into these restricted blocks and would also predict possible entries into the blocks.

Saving Learnt Data Models

In literature, learning patterns from data is usually performed over a prolonged period of time. In order to accomplish that feat the prototype system would have to save the learnt model and then load it and re-use it when needed for more learning or prediction operations. All BN data was saved in XMLBIF format. Furthermore, saving the learnt data would allow off-line analysis and evaluation of the learnt parameters as well as displaying the data visually through JavaBayes [57].

4.8.2 Implementation

This section presents a detailed discussion of all the techniques used to implement the AI component.

Artificial Intelligence Component Overview

The AI component comprised several coherent parts which are shown in Figure 26 on the next page:

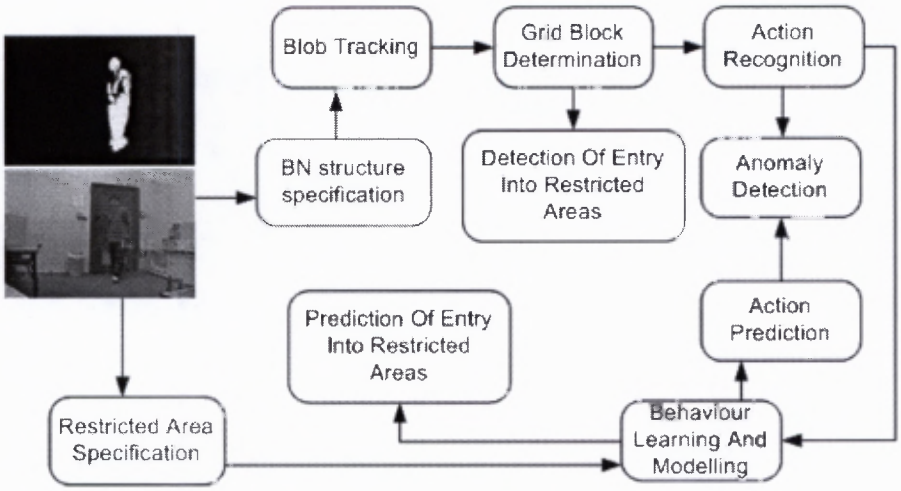


Figure 26: The high-level conceptual view of
the coherent parts that make up the AI component

Each part of the AI component is discussed in detail in the following sections.

Blob Tracking

The first part of the AI component that had to be implemented was the blob tracker. The reason for this is that all the other operations with regard to action recognition and prediction use the object tracker data. At this stage, the IU component had removed noise regions from each frame and formed bounding boxes around the remaining blobs that were assumed to represent the people in the environment. These bounding boxes were used as data by the blob tracking part.

1. Matching Blobs

As mentioned previously an assumption was made that the individuals present in the environment would not be positioned in close proximity to each other. The main reason for this was the fact that the IU could not handle occlusion of people. This assumption allowed a simple proximity based object matching technique, which used the distance covered by each blob's centroid in subsequent frames as an object matching metric, to be used instead of a complex object model. Furthermore, this technique is commonly used in literature for object matching, e.g. it has been used by Bodor *et al.* [2] and Nascimento *et al.* [16].

A list of blobs that were in the environment was kept for reference purposes. In addition, the distance that each blob's centroid moved in successive frames was measured using the Euclidean distance [61]. However a formal evaluation of the distance that each blob moved in

successive frames by a blob's centroid was needed to set a threshold value to use for object matching. This evaluation is discussed in Chapter 5. Briefly, it was discovered that the maximum distance a blob's centroid moved in successive frames was as a result of a running action and the actual pixel based distance was determined to be 150 pixels. Hence, this value was used as an object matching metric.

Thus the inter-frame pixel distance moved by each blob's centroid was computed and analysed in order to match the objects that were being tracked. More precisely, the location of each blob's bounding box centroid in the current frame was compared to all the blobs' centroids that were present in the previous frame. If the distance between the two blob's centroids was less than 150 pixels then the two blobs were matched and assumed to be the same. If a blob in the current frame matched more than one blob in the previous frame then the minimum Euclidean distance amongst all the possible matches was used to resolve the conflict. This technique seemed to perform object matching of the individuals in the environment correctly. An example of the bounding box and Euclidean distance used in the prototype video surveillance system is shown below in Figure 27:

Figure 27: The use of a bounding box and the inter-frame Euclidean distance metric to perform object matching

2. Handling Re-entry into the Environment

It was also discovered that some individuals left the environment for a short period of time and then returned to the environment. This situation had to be handled correctly so that when a person returned to the environment the person's blob would be matched correctly to the blob that left the environment. This was accomplished by analysing the list of current blobs in the environment and allowing a small time period (5 seconds) for the blobs that leave the environment to return. The returning of a blob into the environment was detected by matching the blob's exit block to the blob's entry block. If the blob does not return after the time period it is removed from the current blobs list. This technique is the recommended by Hu *et al.* [5].

Action Recognition

Once the actions that were going to be monitored had been chosen (standing, walking and running), a way of differentiating them had to be established. The logical thing to do was to analyse and monitor the chosen actions so that a common differentiating factor could be found. Upon initial analysis it was evident that the speed of the actions was different. That is a person who is standing was stationary, whilst a person who was walking was moving slowly and a person who was running was moving faster.

This hypothesis had to be verified by analysing the speed, more precisely the displacement, of each action. This was done by analysing the Euclidean distance [61] that each blob's centroid moved in sequential frames. A thorough evaluation of this can be found in Chapter 5. Briefly, the following specifications for the chosen actions were established:

Euclidean Distance – ED – (pixels)	**Action**
$0 < ED \leq 2$	Standing
$2 < ED \leq 40$	Walking
$40 < ED \leq 150$	Running

Table 2: The Euclidean distance metric was used to recognize the action that an individual was performing

Building A BN Representation of the Environment

As previously mentioned a decision was made to divide the monitored environment into a grid of blocks so that positional information of the individuals in the environment could be established. In addition, that decision was made in order to reduce the amount of data that was needed to model the environment. More precisely the environment was divided into a grid of 12 blocks. The next step was to design the BN that would be used to model such an environment.

1. Grid Block Causal Relationships

In order to establish the structure of the BN that would be used to construct a model representation of the chosen environment the actions occurring in that environment were briefly monitored and their block locations were recorded. It became evident that the actions performed by the individuals in a certain grid block were as a direct result of the actions performed by that individual in the neighbouring grid block in the previous time step or frame. This fact suggested that a supervised BN structure could be used and also exposed the causal relationship amongst the neighbouring grid blocks which is shown below. In Figure 28 below, Block 5 is used as a reference block and the black directional lines show the causal relationships for that block.

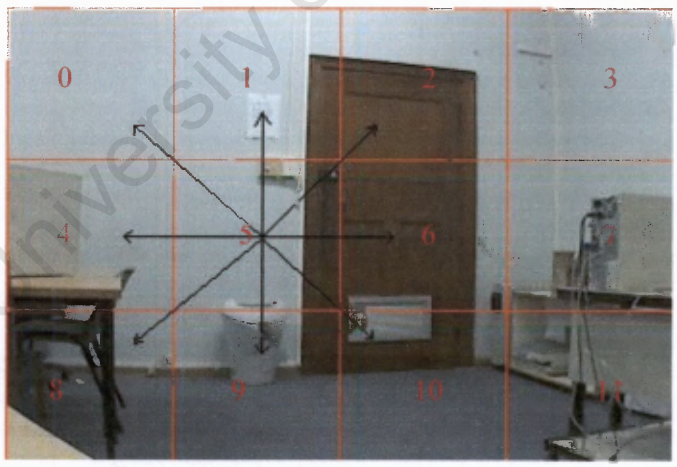


Figure 28: The grid block causal relationships and numbers

Hence, it was initially decided that a single BN would be used to model such a grid environment with each block being a node in the BN and the actions that could happen within the block being the states of the each node. However, the fact that a BN cannot have bi-

directional causal relationships, e.g. actions in block 5 directly influence the actions in block 6 and vice versa, resulted in a re-analysis of the initial BN structure.

2. *Different BN for Each Grid Block*

To overcome the previously mentioned bi-directional relationships amongst neighbouring blocks it was decided that each block in the grid would be modelled using a different BN. In such a way bi-directional causal relationships could be split into two single causal relationships between the neighbouring blocks. Thus a collection of different BNs was used to model the environment. Implicitly this decision meant that the large amount of data that would have been used to model the environment using a single BN could be distributed amongst the 12 BNs used to model each block of the environment grid. This type of BN structure specification is based on the heuristic knowledge about the environment and the relationships between the actions in the grid blocks. Figure 29 below shows the general BN structure used to model each grid block:

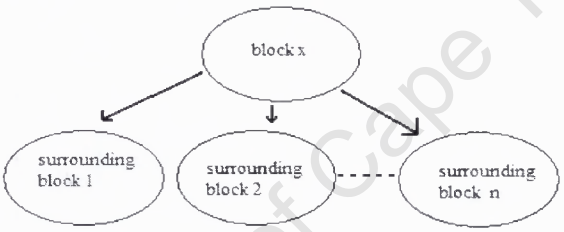


Figure 29: The General BN that was used to model each block in the environment grid

3. *Avoiding Cycles*

A further analysis of the behaviours in each block exposed the fact that the BN structure had cycles present which contradicts the fact that a BN cannot have cycles. This was mainly caused by the fact that the size of the grid blocks was larger than the individuals in the environment. Hence an individual would remain in a single grid block for a prolonged period of time. In order to handle this type of activity and remove cycles from the BN a special node was introduced into each BN to represent the fact that an individual can remain in the current block in a future time. This node was denoted as *currentBlock* with an *s* at the end, e.g. block0s. This can be seen in Figure 30, on the next page, where *pos5s* represents remaining in *pos5*.

4. Final BN structure

After all the above points were put together the final supervised, or static, BN structure was established. The BN structure and CPT structure for the same reference block 5 used previously are shown in Figures 30 and 31 below, using JavaBayes [57] as a graphical interface to view the BN.

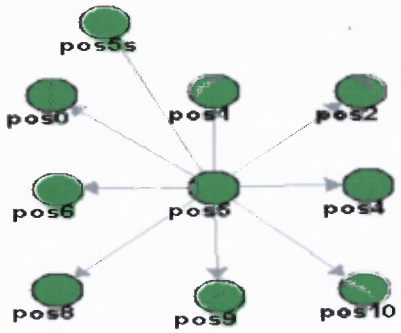


Figure 30: The final supervised BN structure for block 5

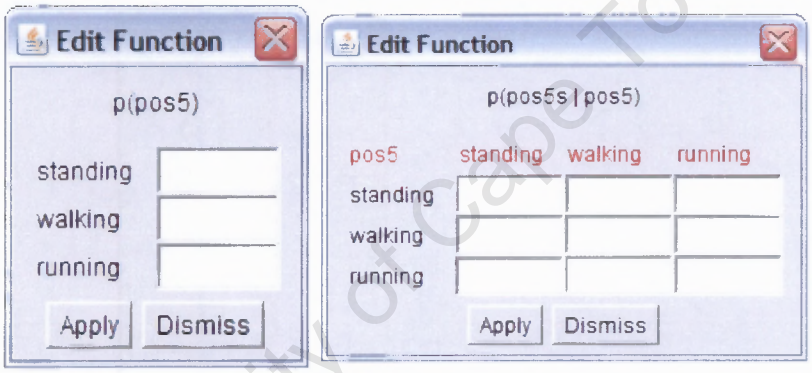


Figure 31: The CPT table for block on the left and
the CPT table for block 5s given block 5 on the right

Finally, BaBe [62] was used to build all the BNs that were used in the prototype video surveillance system and it was also used to perform all the inference tasks. To avoid confusion the name of the BN for each grid block started with the word *block*, whilst the node names started with word *pos*. The use of a combination of BNs to model the grid environment provided the missing pieces for the solution to the second research question and also set up a framework that could be evaluated in order to answer the first, third and fourth research questions.

BN Parameter Learning

Once the BN structure had been established the next step was to learn behaviour patterns in the environment so that predictions based on those learnt behaviour patterns could be made. This was accomplished by performing parameter learning for each BN in order to populate the CPT tables of each node and to complete each BN's specification. The prototype surveillance system was designed to have two operation modes, namely *learning mode* and *live mode* (discussed in the following section), which could be specified through the prototype user interface. An evaluation of the learnt data is presented in the next Chapter.

1. Learning Mode

An initial learning period was needed to populate each node's CPT table in every BN. The fact that the structure of each BN was known and the data for each environment grid block could be obtained made the learning process simple. From the discussion presented in Chapter 2 about BN learning it is clear that the learning technique required for this situation, where the BN structure is known and the data variables are observable, is the MLE algorithm. The MLE in this case was implemented as an occurrence count. In addition, this type of learning can be classified as semi-supervised data learning. Finally, a decision was made to assign a default walking action to a blob when it first entered the environment.

2. Use of Learnt Data

Although each grid block had its own BN all the neighbouring grid blocks had global causal relationships between them. This meant that the learning data for each block had to be used in some of the neighbouring blocks BN as well. Hence the BN for the block where a blob was located was updated by the learning algorithm as well as the neighbouring block that caused the action in the current block. To demonstrate this fact the following example is used.

Scenario:

- Frame 1: Blob enters in block 4
- Frame 2: Blob walks in block 4
- Frame 3: Blob walks in block 5
- Frame 4: Blob stands in block 5
- Frame 5: Blob runs to block 4

When the blob initially enters into block 4, in frame 1, the learning algorithm updates the BN for block 4 by incrementing the walking parameter, $P\ (pos4 = walking)$, in the node *pos4's* CPT. In frame 2, the next action the blob does is to remain in block 4 whilst walking. Again the BN that would be updated would be block 4's BN. However, in this case two different node's CPT tables had to be updated. Firstly the fact that the blob was walking in block 4 meant the walking parameter, $P\ (pos4 = walking)$, in node *pos4's* CPT had to be incremented again. Secondly, the fact that the action that led to the blob walking in block 4 in frame 2 was walking in block 4 in frame 1 had to be recorded. Hence, the parameter $P\ (pos4s = walking \mid pos4 = walking)$ in node *pos4s'* CPT also had to be updated to represent that fact. This is an example of how cycles were avoided in the BN structure specification.

In frame 3 the blob was walking in block 5. This meant that the BN for block 5 had to be updated to represent that action. Hence, the parameter $P\ (pos5 = walking)$ in node *pos5*'s CPT is incremented. At the same time, the fact that walking in block 4 led to walking in block 5 had to be updated as well. Hence an update was done to block 4's BN. The parameter $P(pos5 = walking \mid pos4 = walking)$ in node *pos5's* CPT table was incremented.

In frame 4, the blob stood in block 5. This meant that the BN for block 5 would be updated by incrementing the parameter $P\ (pos5 = standing)$ in node *pos5*'s CPT. At the same time, the parameter $P\ (pos5s = walking \mid pos5 = standing)$ in node *pos5s*' CPT was incremented. Finally in frame 5, the blob ran in block 4. Similarly this meant that the BN for block 4 had to be updated by incrementing the parameter $P\ (pos4 = running)$ in node *pos4*'s CPT. At the same time the BN in block 5 had to be updated to represent the causal relationship that happened. Hence, the parameter $P\ (pos4 = running \mid pos\ 5 = standing)$ in node *pos4*'s CPT was incremented as well.

3. Saving and Loading the Learnt Data

In order to minimize the effects of the problems associated with the MLE algorithm with regards to scarce amounts of data, the learning of environment data was done over a prolonged period of time in order to maximise the amount of learnt data. Hence, the learnt data had to be saved for re-use at later stages and also in order to avoid data loss. Thus, every time the prototype video surveillance was switched off the collection of BNs that was being

used was stored in an XMLBIF file. Likewise, when the prototype video surveillance system was started, a set of BNs had to be selected via the prototype user interface before any surveillance tasks could be done. This decision also allowed the use of various sets of BNs.

Action Prediction (Live Mode)

Once a reasonable amount of environment data had been learnt, action prediction could be performed. This required the prototype video surveillance system to be running in *live mode* which allowed prediction of actions to be done. In addition, during *live mode* learning of the environment data was still being done to maximize the amount of learning data and also to capture new action trends or patterns. Furthermore, BaBe's variable elimination algorithm was used to perform the inference on each block's BN.

Action prediction was done by using the BN of the grid block, were the blob was located, to find all the probabilities of the blob's possible future action. Each node in the BN was sequentially queried to get the probabilities of possible future actions based on the blob's current block position and action. For example, if a blob was positioned in block 5 and was walking in the current frame, then the BN for block 5 was used to get the probabilities of the blob being in all the neighbouring blocks in the next frame. A possible prediction was worked out for all 3 actions in the neighbouring blocks. The fact that the blob was in block 5 and was walking was used as evidence during the queries. Hence all the actions in all the neighbouring blocks were queried with $P\ (pos5 = walking)$ being the evidence.

Instead of making a single prediction of the next possible action for each blob, a decision was made to use a list of possible actions. Hence all possible future action probabilities that were larger than 0% were stored in the action probability list. This decision gave a broader probability range for action prediction.

Prediction of movement into a restricted block was done by firstly storing a list of user defined restricted blocks which were specified through the prototype user interface. Then each predicted action and block position pair were compared to that list. If the predicted action was in a block that was in the restricted list then predicted movement into a restricted block was detected and a signal was raised.

An evaluation of the inference values that were obtained by the prototype video surveillance system is presented in the next Chapter. In addition, action prediction was performed in near-real time due to the fact that the size of each BN was not large. Near-real time performance was easy to detect because of the nature of the prototype system. There was only a slight lag time between the action happening in real life and the actions being displayed by the prototype system. Hence, the fourth research question regarding near-real time action prediction was answered by using smaller sized BNs to minimize the processing needed during inference.

Anomaly Detection and Prediction

As mentioned previously anomaly detection was done by comparing blob actions to learnt action patterns in addition to the use of heuristic knowledge. More specifically, when a blob's action in a certain block was recognized it was compared to that blob's learnt action patterns in order to establish whether the action was an anomaly or not. A decision was made to classify any action with a common pattern probability of less than 20% as an anomaly. As an example, if a blob is walking in block 5, then the BN for block 5 would be queried to get the learnt probability pattern of walking in that block i.e. $P(pos5 = walking)$. If that probability was less that 20% then an alarm was raised. Furthermore, a list of heuristically specified anomalous actions was kept. If the current action and block pair were in the heuristic list then the action would be considered as an anomaly regardless of the learnt action patterns by the BN. This demonstrated a way to override the BN anomaly detection system.

In addition to using the current block's learnt behaviour patterns. The block where the blob was in the previous frame was also used to detect anomalous action. As an example a blob can be assumed to be walking in block 5 in the current frame whilst it was running in block 6 in the previous frame. Then the BN in block 6 would be queried to determine the probability that $P(pos5 = walking \mid pos6 = running)$. If this probability was less than 20% then the walking action in block 5 would be regarded as an anomaly based on the learnt behaviour patterns of the BN in block 6.

Anomalous action prediction was done in a similar way to anomaly detection. When a prediction was made about a future action, the BN in the block where the action would

happen was queried to find out the common behaviour patterns in that block. As an example, if the predicted action was standing in block 8, then the BN in block 8 would be queried to find out the common action patterns in that block. In other words, the BN in block 8 was queried to get the probability $P\,(pos8 = standing)$. If this probability was less than 20% then the predicted action was regarded as a predicted anomalous action.

The detection and prediction of anomalous behaviour relied on the accuracy of the inference algorithm. Finally, by detecting anomalies in the ways discussed above the third research question regarding anomaly detection based on learnt behaviour patterns was answered.

4.9 Prototype Interface Screen Shot

The following screen shot, Figure 32, shows the user interface of the prototype video surveillance system.

Figure 32: The user interface for the prototype surveillance system

4.10 Prototype Evaluation

The final step in the research methodology was to evaluate the overall accuracy, robustness and efficiency of the prototype surveillance system. The prototype video surveillance system

saved several different images and also kept logs of the processes it was performing for evaluation purposes. The evaluation of the prototype is detailed in the next Chapter. In addition, although the prototype was developed for a specific environment some of the PETS 2002 data sets [41] were used to test the functionality of the prototype.

4.11 Conclusion

This Chapter showed how the IU and AI techniques presented in Chapter 2 were used to build the prototype video surveillance system that demonstrated the ideas and research goals presented during this dissertation. A clear indication of the research focus was presented at the start of the Chapter. The design decisions and implementation details of each component of the prototype video surveillance system followed.

Briefly, the IU component was built as a pluggable data capturing and information gathering component that worked on image sequences received from an Axis 210 camera. The output of the IU component was a collection of bounding boxes that represented the individuals in the environment. This collection of bounding boxes was then processed by the AI component in order to understand what behaviours were happening in the environment, learn the common behaviour patterns and use those patterns to predict future actions. A collection of BNs was used at the heart of the AI component to learn the behaviours in the environment. In addition, these actions were analysed in order to detect and predict anomalous behaviours.

The next Chapter discusses the design and implementation evaluation of the prototype video surveillance system that was carried out.

Chapter 5

Evaluation and Results

This Chapter describes the evaluation that was performed on the prototype video surveillance system and presents the evaluation results that were obtained. The first half of the Chapter is dedicated to design time evaluations that were necessary for the implementation of the prototype system. The second half of the Chapter is dedicated to the overall prototype system evaluation, including performance, robustness and accuracy evaluations. In addition, for comparison purposes a well established indoor surveillance data set, the PETS 2002 dataset 1 [41], was used to test the functionality of the prototype system for comparison purposes with other researchers.

5.1 Evaluation Hardware

The majority of the prototype video surveillance system evaluation was performed on an Intel Pentium 4 1.79 GHz Computer with 512 RAM. However, some of the final evaluation tests were run on a computer with an Intel duo-core T2300 1.66 GHz processor and 1 GB RAM. The operating system on both computers is Windows XP Professional and the development environment is the C# Microsoft Visual Studio development environment. It should be assumed that the computer used for any particular evaluation test is the 1.79 GHz Pentium 4 unless otherwise specified.

5.2 Design Time Evaluation

This section of the Chapter focuses on the design time evaluation that was performed in order to make vital design decisions during the development of the prototype video surveillance system. Each evaluation test was run on a 300 frame sequence, approximately 40 seconds of actual video footage. In addition each test was repeated 5 times to minimize the effect of errors in the evaluation.

5.2.1 The Euclidean Distance for Merging Small Blobs

The distance used to merge the small blobs that belonged to a larger blob which represented the individual in the environment had to be determined through an evaluation of the

Euclidean Distance [61] between the small blobs that were in close proximity. The evaluation procedure was as follows:

- Store a reference image of the individual in the environment. This will be used as ground truth in determining which small blobs need to be merged;
- Store the binary image of the blobs in the environment;
- Label each small blob in the binary image and calculate the Euclidean Distance between each of the small blobs. A special note must be added that small blobs that were within the bounding box of another blob were automatically merged and were given a Euclidean distance of 0;
- Make a visual comparison of the location of each small blob and the location of the individual in the environment to determine whether the blobs need to be merged. By so doing a threshold Euclidean distance can be evaluated and used to determine whether to merge closely located small blobs;
- Repeat for 5 tests;

The average of the maximum valid Euclidean distance found by the 5 tests was used as the threshold Euclidean distance. The detailed results of all the evaluation tests can be found in Appendix B, section B.1. A representative sample of the evaluation results is represented by evaluation test 1 which is presented below in Figure 33 and Table 2:

Test 1 Results

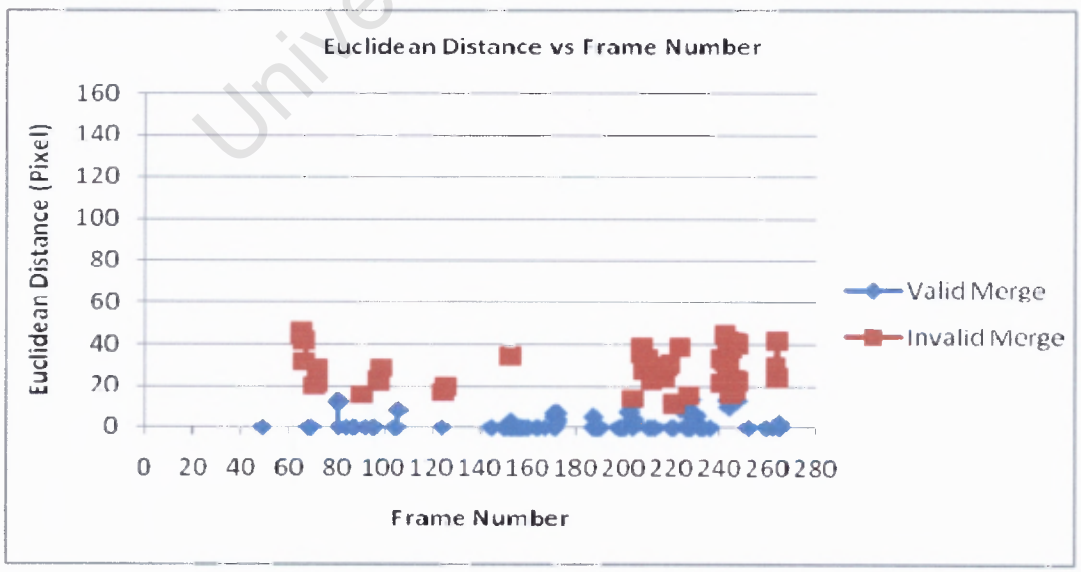


Figure 33: The Euclidean distances for valid and invalid small blob merge operations for test 1

Euclidean Distance (Pixel)	Valid (Pixels)	Invalid (Pixels)
Maximum	23.6	46.7
Minimum	0	11.1
Average	2.1	28.9

Table 2: The summary of merging small blobs test 1 results

Analysis of Tests

After analysing the 5 evaluation tests it was clear that a threshold value could be established in order to merge small blobs that were in close proximity and that belonged to the same foreground individual. In Figure 33, on the previous page, one can see that the valid merge operations are generally below the 20 pixel Euclidean distance line whilst the invalid merge operations are above that line. This was a common trend in all the tests that were conducted. In addition, this trend is supported by the vast difference between the average Euclidean distances of the two operations, shown in Table 3 above.

In addition, by analysing Figure 33, on the previous page, one can see that a few valid merge operation Euclidean distances fall within the invalid merge range and vice versa. These values could be considered to be outliers. Many researchers omit these values from any statistical analysis. A decision was made to use the average maximum valid merge Euclidean distance as a metric for determining whether to merge small blobs in close proximity.

Hence, the average maximum valid Euclidean distance for valid merge operations was found to be $(23.6 + 24.7 + 33.1 + 27.5 + 24.6)/5 = 26.7$. A further decision was made to round down this value to 20. This was done in order to get a threshold that would be easier to use. Such a decision was justified by the fact that the average valid merge Euclidean distance over all the tests was 4.3 pixels, whilst the average invalid merge Euclidean distance was 74.9 pixels. If any small blobs were not correctly merged they would be filtered out as larger noise regions by latter steps in the IU component.

5.2.2 The Size of Large Noise Regions

The next design evaluation that had to be performed was the determination of the size of the larger noise regions that were present in the environment data. A manual evaluation, similar to the one used for determining the merging Euclidean distance between small blobs, was used to determine the size of the noise regions. The evaluation procedure was as follows:

- Store a reference image of the individual in the environment. This will be used as ground truth in determining which blobs represented the individual and which represented the larger noise regions;
- Store the binary image after the second pass of the connected components algorithm in order to get the full blob representation;
- Label each blob in the binary image for identification purposes and extract the dimensions of the blob's bounding box;
- Make a visual comparison of the location of blob in the reference image and the binary image in order to determine which dimensions correspond to the larger noise regions. By doing so a threshold dimension could be established that could be used to filter out the size of the larger noise regions;
- Repeat for 5 tests.

The average of the dimensions found by the 5 tests was used as the threshold dimensions for filtering larger noise regions. The detailed results of all the evaluation tests can be found in Appendix B, section B.2. A representative sample of the evaluation results is represented by evaluation test 2 which is presented below in Figure 34:

Test 2 Results

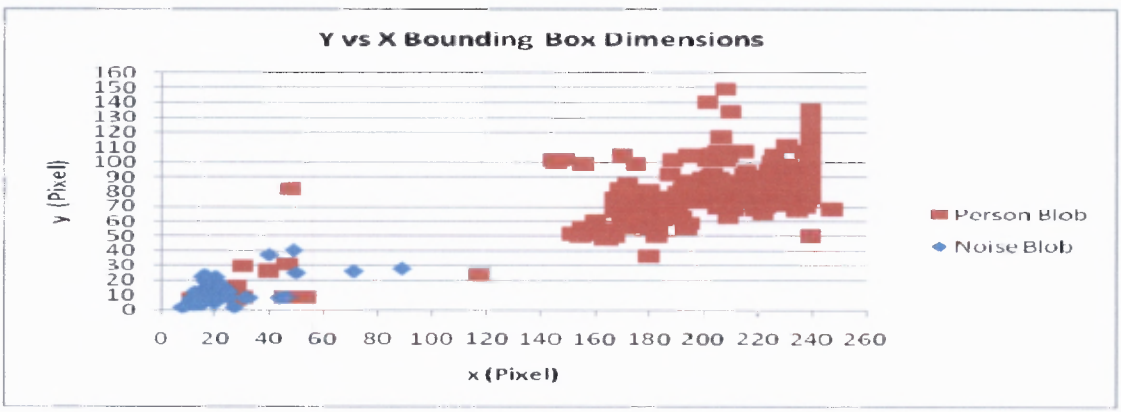


Figure 34: The dimension clusters for noise regions and person blobs for test 2

The average dimension of the noise regions' bounding box for Test 2 is 18.1 x 8.9.

Analysis of Tests

As demonstrated by the sample evaluation test 2 shown in Figure 34, on the previous page, there are 2 clear groupings or clusters that represent the majority of data points for large noise regions and individuals in the environment. This was a common trend found in all the evaluation tests. However, in test 2 there appeared to be noise regions that were larger than the common cluster size. After further analysis of the test images it was discovered that this was as a result of shadows in the environment. In addition some of the individual blob sizes are much smaller than the cluster suggests. Likewise, after analysis of the images it became evident that this was as a result of occlusion of the individual in the environment.

Hence, taking the average dimensions over the 5 tests the following average noise region dimensions were obtained: 16.2 x 8.5 pixels. In order to capture the outlying dimensions of any noise regions, the threshold dimension used for filter noise regions was 20 x 20 pixels. This threshold dimension filter proved to remove the majority of the larger noise regions.

5.2.3 Object Tracking Euclidean Distance

In order to track an individual in the environment a threshold had to be established for the distance moved between successive frames. This threshold would allow object matching to be done and would also help in determining new blobs in the environment. The evaluation procedure was as follows:

- Store a reference image of the individual in the environment. This will be used as ground truth in determining which blobs represented the individual and which represented the larger noise regions;
- To simplify the experiment and provide clarity only one individual performed actions in the environment so that a threshold Euclidean distance for matching and tracking the individual could be established;
- Calculate the inter-frame Euclidean distance for successive frames;
- Verify that only one blob is present in each frame;
- Repeat for 5 tests.

The average of the Euclidean distance found by the 5 tests was used as the threshold Euclidean distance which was used as an object matching and tracking threshold. The results of the evaluation tests were combined into one graph and are presented in Figure 35 below:

Test Results

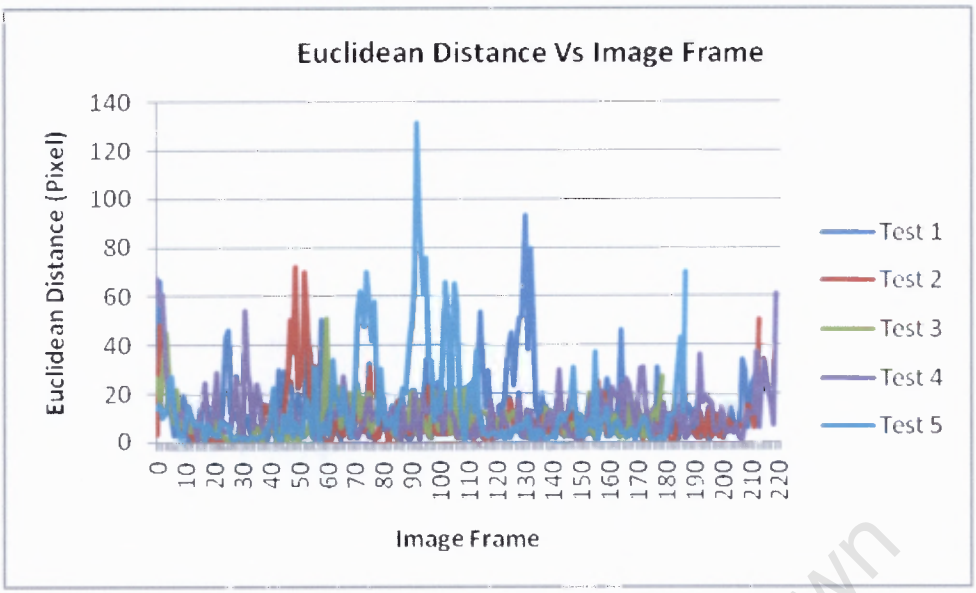


Figure 35: The results of the object matching and tracking Euclidean distance evaluation

Analysis of Test Results

From the Figure 35, above, it can be seen that the average inter-frame Euclidean distance that a blob moves between successive frames is well below 80 pixels. However, test 5 shows a high spike of Euclidean distance going up to 130 pixels. This was caused by the fact that the running action was the most common action performed during test 5. Hence, the Euclidean distance threshold value for matching and tracking blobs in the environment was chosen to be 150 pixels so that any outlying running Euclidean distance spikes could be correctly matched to the same blob. This value seemed to adequately match all the blobs in the environment.

5.2.4 Action Identification Euclidean Distance Metric

In order to recognize the type of action that is happening in the environment a common characteristic had to be used to differentiate the various actions, namely standing, walking and running. This characteristic was determined to be the inter-frame displacement of a blob between successive frames. However, an evaluation of the various displacement values for each action had to be done to specify the threshold range for each action. The evaluation procedure was as follows:

- Store a reference image of the individual in the environment. This will be used as ground truth for ensuring that only one blob was being tracked during the evaluation;

- The individual in the environment performed only one type of action in each test;
- The inter-frame Euclidean distance moved by the bounding box centroid of the individual for successive frames was calculated ;
- Repeat for 5 tests for each type of action.

Intuitively standing should have the smallest displacement, whilst running should have the largest displacement. The results of the average of the 5 evaluation tests for each action are presented below:

Average Test Results

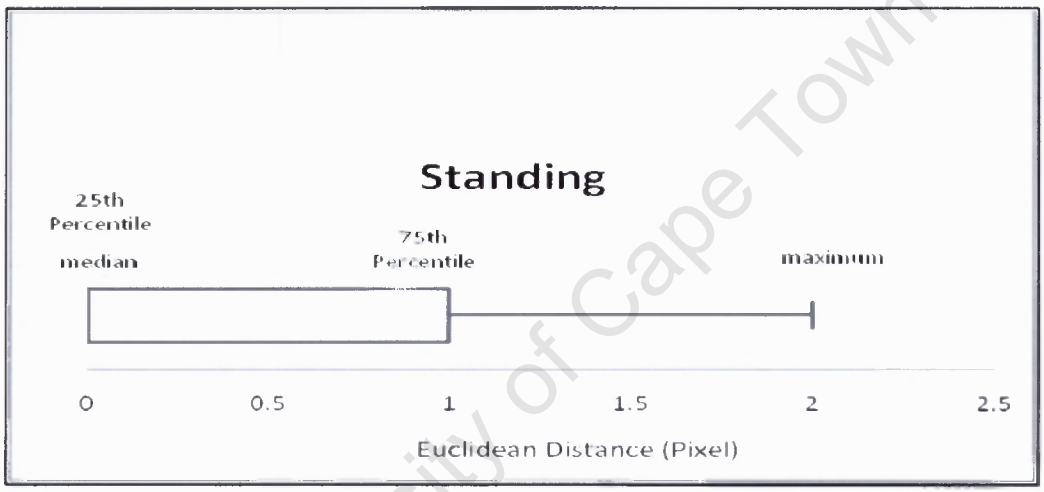


Figure 36: Box and whisker plot for standing

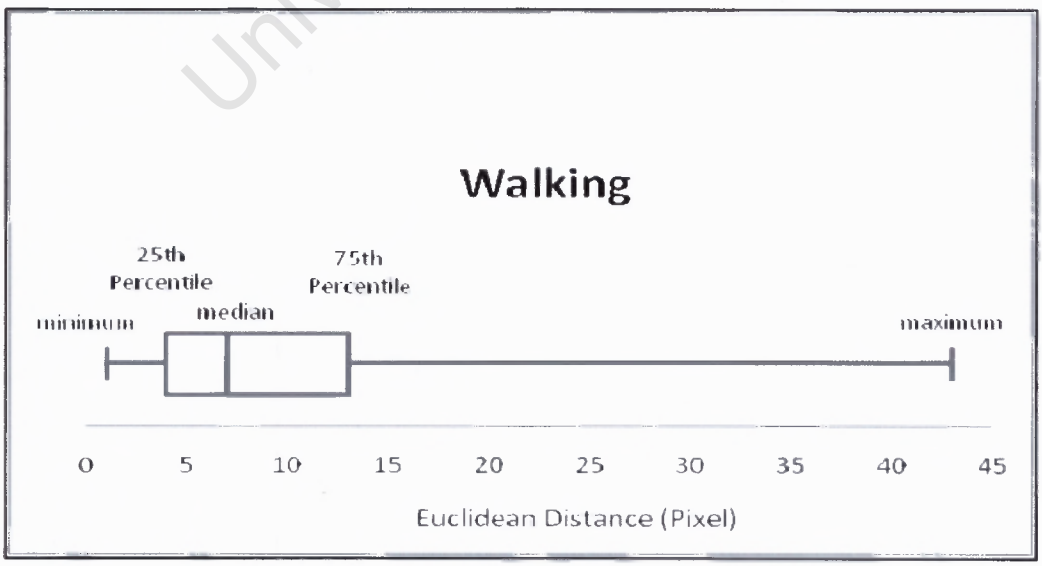


Figure 37: Box and whisker plot for walking

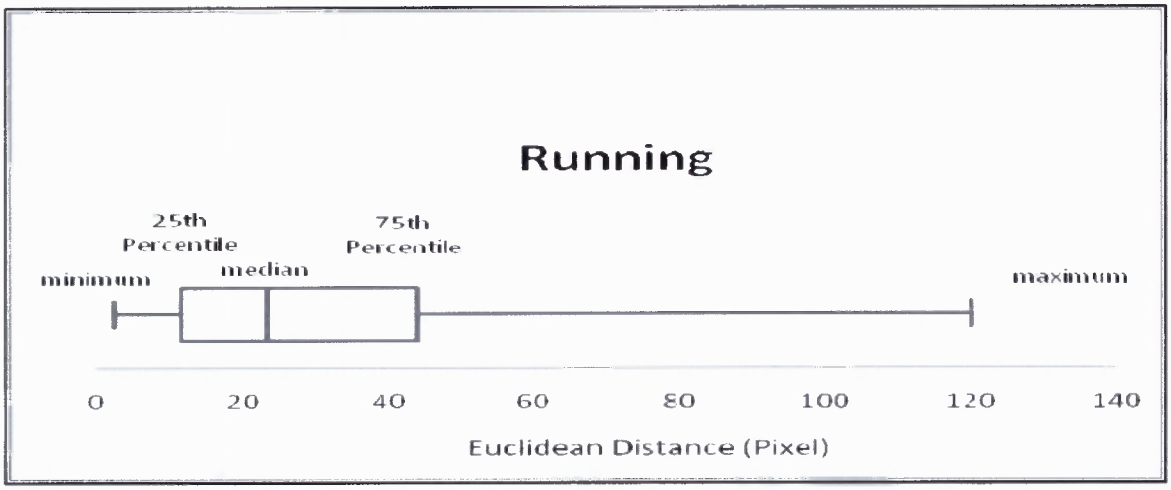


Figure 38: Box and whisker plot for running

Analysis of Results

By analysing Figures 36, 37 and 38, on the previous page, it can be seen that the three actions can be identified by the tight clusters that form around each action's median Euclidean distance displacement value. One has to note that the running action in Figure 38 seems to have median and minimum values within the range of the walking action. This is as a result of several walking actions happening in between the running actions. In addition, each action has outlying values that form a fuzzy area around the borders of each action's identifying Euclidean distance displacement ranges.

One important fact that was revealed by analysing the images of the evaluation tests is that when an individual is performing an action in the line of view of the camera the action can not be distinguished from the standing action. However, when the actions are performed across the field of view of the camera then they can be clearly identified. This problem could not be addressed by the prototype video surveillance system because the camera that was used could not detect the depth of the objects in its field of view.

Hence, a heuristic decision was made to identify the borders of each action's Euclidean distance range. Standing would be identified by a Euclidean distance displacement that was less than 2 pixels. Walking would be identified by the range between 2 pixels and 40 pixels. Running would be identified by any Euclidean distance displacement that was greater than 40 pixels and less than 150 pixels. 150 pixels was the threshold metric used to match tracked objects in subsequent frames.

5.3 Overall Prototype Evaluation

This section of the Chapter focuses on the overall prototype video surveillance system evaluation that was performed. The accuracy of the inference values calculated by BaBe [62] were evaluated by a comparison to JavaBayes [57]. The robustness and accuracy of several parts of the prototype surveillance system were evaluated through analysing test results and comparing them to ground truth data. Finally, the prototype surveillance system was run on a common dataset, namely PETS dataset 1 [41] in order to perform a comparison with other researchers.

5.3.1 Evaluation of Inference

The inference results obtained from BaBe [62] were used as the basis for predicting the action of an individual in the next frame or time step. In addition, the inference values were also used for determining anomalous actions. Hence an evaluation of the inference values that were obtained was needed in order to make sure that accurate inference was being performed and used in determining future actions as well as detecting anomalies. This was done by comparing the values calculated by BaBe [62] to the values calculated by JavaBayes [57] for specific scenarios. A similar BN to the BN that was used previously in the dissertation, namely the BN for grid block 5 with learnt parameter values, was used for the inference test.

The exact evaluation procedure is presented below:

- The BN was specified so that it could be used in both JavaBayes [57] and BaBe [62];
- *Pos5* was set to standing, which was used as evidence;
- The inference values for nodes *pos4*, *pos6* and *pos5s* were noted;
- Repeat for *pos5* set to walking and then running.

The CPT parameter details of the BN that was set-up are shown in Appendix B, section B.5. The structure of the BN is shown in Figure 39 below:

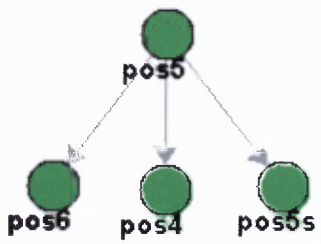


Figure 39: The BN used for inference evaluation

Test Results

The actual results of the inference evaluation test can be found in Appendix B. Section B.3. A representative sample of the evaluation tests is test 1 which is shown below in Figure 40:

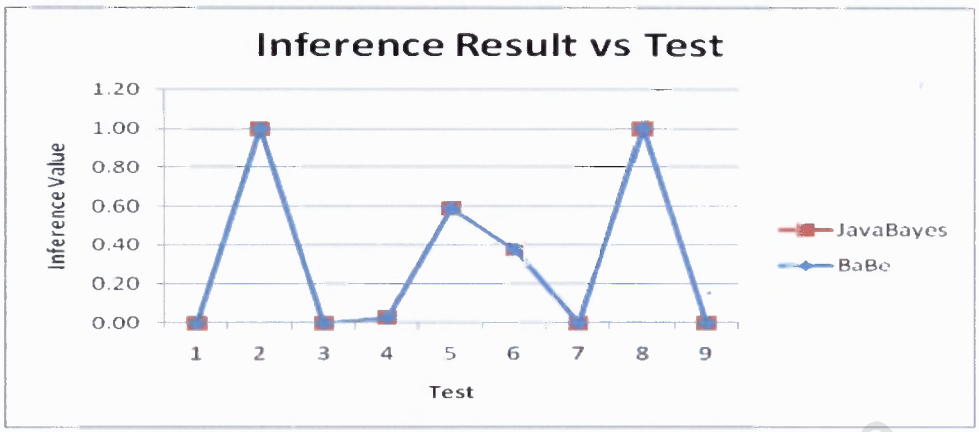


Figure 40: Inference Test 1 Result

Analysis of Results

By analysing Figure 40, above, it can be seen that the inference values obtained by BaBe [62] and JavaBayes [57] were virtually identical. This fact was common for all the evaluation tests that were performed. Hence, it can be concluded that satisfactory inference results were obtained by BaBe [62]. In addition, the action predictions were as accurate as the inference values obtained by BaBe [62] and JavaBayes [57].

5.3.2 Evaluation of Performance

Although the prototype video surveillance system managed to achieve near-real time performance, with only a slight time lag, a comparison with some of the existing surveillance systems had to be performed to justify such a performance. The performance of a surveillance system can be measured by looking at several aspects, including the analysis of the maximum frame rate that it can handle.

When saving reference images for evaluation purposes and using the development computer, the prototype video surveillance system achieved a frame rate of 7 frames per second (fps) with an image resolution of 352 x 240 pixels with a fair amount of lag between the frames it was processing and what was actually happening. However, when the reference images are not saved the amount of lag is considerably reduced. 7 fps proved to allow smooth inter-frame movement to be achieved without any jerking or stuttering effects. In addition, the

chosen image resolution provided a clear image with good visibility. Real-time performance could be achieved by lowering the frame rate, but that would result in jerking or stuttering effects in the inter-frame movement of the individuals. In addition, one could lower the image resolution to achieve better performance at the expense of image quality.

On the Intel duo-core T2300 computer the frame rate of 7 fps resulted in smooth real-time performance. In addition, a frame rate of 14 fps resulted in only a small lag time. However, the lag time was dependent on the amount of foreground pixels. Although real-time performance was achieved on a faster computer, the whole prototype video surveillance could be optimised to achieve better performance on a standard computer.

In comparison, Stauffer and Grimson [4] achieved a performance of 11-13 fps at a lower image resolution of 160 x 120 pixels. Lipton *et al.* [7] achieved a performance of 14 fps at an image resolution of 320 x 240 on a weaker computer. However, their system did not have the capability to predict actions or detect anomalies. Cucchiara *et al.* [8] achieved a frame rate of 15 fps on a standard computer, but they do not reveal the image resolution that they used. Ivanov *et al.* [10] achieve a frame rate of 12 fps on a lower image resolution of 120 x 160 pixels. The video surveillance systems used for comparison purposes form only a small subset of all the video surveillance systems that have been developed over the years. They probably do not exhibit the best performance but instead represent some of the surveillance systems that have been published in literature.

The frame-rate and the near-real time performance that was achieved by the prototype video surveillance system helped to answer part of the last research question regarding near-real time predictions of human behaviour.

5.3.3 Evaluation of Robustness and Accuracy

This section focuses on the evaluation tests that were performed in order to determine the robustness and accuracy of several parts of the prototype video surveillance system. To ensure the validity and integrity of the results that were obtained during the evaluation, a volunteer was asked to perform the required actions in the environment during the evaluation procedure. Some of the evaluation metrics that were presented in Chapter 2 were used to

evaluate the prototype video surveillance system. The following areas were evaluated in each evaluation test:

- Blob count – to demonstrate the ability to remove shadows and noise regions. This should always be 1 because only one individual was involved in the evaluation tests;
- Tracking of the blob - to demonstrate the ability to track the correct individual;
- Action recognition – to demonstrate the correct recognition of the actions that the volunteer was performing;
- Action prediction – to demonstrate the accuracy of the predictions;
- Anomaly detection and prediction – to demonstrate the ability of the prototype surveillance system to detect and predict anomalous behaviour;
- Detection and prediction of entry into restricted areas.

A decision was made not to evaluate the accuracy of the blob segmentation process because it was assumed that as long as a part of the volunteer was segmented then correct tracking could be done. In addition, only one volunteer was involved in the evaluation tests so partial segmentation of the object would suffice. Blob classification was implicitly covered by the blob count evaluation because at all times there should only be one person in the environment.

The following evaluation procedure was used:

- Ask the volunteer to perform actions that they felt like doing for a specified time period which would be used as training data for the volunteer's BN;
- Observe the common patterns of the volunteer during the training phase so that specific evaluation test procedures could be made;
- Ask the volunteer to perform different tasks in order to establish the robustness and accuracy of the prototype video surveillance system;
- Use both the BN trained on the volunteer's actions and the BN trained over 3 months of all individuals actions;
- Keep reference images of the volunteer in the environment that will be used to extract ground truth data;
- Store the output of the video surveillance system so that a comparison with the ground truth data can be made.

An example of the evaluation procedure is presented below:

Scenario: Block 7 is a restricted area block. The blob is walking in block 6 in the current frame. This is a normal action in that block. In the next frame the blob will walk into block 7. This should be detected as entry into a restricted block. In the last frame the blob will be standing in block 7, which is an anomalous activity based on that block's training data. The table below shows the output of the prototype system for those frames:

Frame: 1	**Frame: 2**	**Frame: 3**
Matched Object: 0	Matched Object: 0	Matched Object: 0
Block: pos6 State: walking	Block: pos7 State: walking	Block: pos7 State: standing
Current Action Likelihood: 83%	Current Action Likelihood: 83%	Current Action Likelihood: 16% (anomalous action detected)
Predicted movement: pos5 walking 25%	Restricted Block Detection: Block 7	Restricted Block Detection: Block 7
Predicted movement: pos7 walking 23% (disallowed block)	Predicted movement: pos6 standing 3% (anomaly, action likelihood 15%)	Predicted movement: pos6 walking 50%
Predicted movement: pos7 running 2% (disallowed block) (anomaly, action likelihood 1%)	Predicted movement: pos6 walking 31%	Predicted movement: pos7s standing 19% (disallowed block) (anomaly, action likelihood 16%)
Predicted movement: pos10 walking 25%	Predicted movement: pos11 walking: 33%	Predicted movement: pos7s walking: 31% (disallowed block)
Predicted movement: pos6s standing 3% (anomaly, action likelihood 15%)	Predicted movement: pos7s standing 4% (disallowed block) (anomaly, action likelihood 16%)	
Predicted movement: pos6s walking 21%	Predicted movement: pos7s walking 29% (disallowed block)	
Predicted movement: pos6s running 1% (anomaly, action likelihood 2%)		

Table 3: Example evaluation scenario results

Referring to Table 3, on the previous page, blob counting, blob tracking, action recognition, anomaly detection and restricted block entry detection are straight forward to evaluate because they are true or false type of evaluations, e.g. in the first row of the table, blob 0 was matched in all three frames so tracking and blob count are correct. However, when it comes to all the predictions, a bit of thought was needed to set up the correct evaluation procedure.

When training data is used, predictions can only be made based on the learnt training data patterns. In addition, predicting anomalies is a very difficult thing to do because anomalies are not common behaviour patterns. Thus, for action prediction the top 3 action predictions were considered in the evaluation. The main reason for doing this is that a set of predictions is generally more accurate that a single valued prediction. From Table 3, on the previous page, the top 3 action predictions in Frame 1 were: *pos10* walking 25%, *pos5* running 23% and *pos7* walking 23%. Thus, one can conclude that the predicted action for Frame 2 was correct.

However, for anomaly predictions the top 3 predicted actions will very seldom result in correct anomalies in the next frame because they are the common behaviour patterns that the BN has learnt whilst anomalies are patterns that are not common. In fact, if a BN has never come across a certain action it will never predict its occurrence. Hence, a decision was made to consider all possible anomaly predictions when determining if an anomaly was correctly predicted. From Table 3, on the previous page, an example of the anomaly prediction evaluation would be in Frame 2 the predicted action *pos7* standing has a low percentage of 4.4% and does not fit into the top 3 action predictions. However, the action prediction is detected as an anomalous prediction, which turns out to be true in Frame 3. If only the top 3 predictions were considered such an anomaly prediction would go unnoticed.

Unfortunately, this decision can result in lots of false alarms being raised. Thus, in determining a false anomaly prediction only the top 3 predictions were considered. In other words, if an action is predicted with a high percentage of being an anomaly and it did not happen in the next frame then this is a definite false alarm. These decisions gave a nice balance to the evaluation of the anomaly prediction procedure. The same procedure applied to predicting movement into a disallowed block.

BN Collections

During these evaluation tests, two different collections of BNs were used to see the difference between individual and environmental learnt behaviour pattern. The first BN collection was trained using the volunteer's actions during the training period, which is discussed later. The second BN collection was trained over a 3 month period and learnt the common patterns in the Agents Research Lab of all individuals.

Training Period

In literature, researchers commonly refer to training periods during which their surveillance systems learn the common behaviour patterns in the environment. This approach was used during the robustness and accuracy evaluations tests. The training period for the BN that learnt the common patterns of the volunteer was 5 minutes (approx. 2100 frames). This training period was similar to the training period used by Lazarevic-McManus *et al.* [42], which was 5 ½ minutes. In addition, this training period was longer than the three PETS 2002 [41] training data sets which were 377, 1471 and 1295 frames respectively. Hence, the duration of the training period was justified because it was similar to the training periods used by several other researchers.

Ground Truth Data

The ground truth data for each evaluation test was extracted manually from the reference images that were stored by the prototype video surveillance system. This ground truth data was then used to establish the robustness and accuracy of several parts of the prototype video surveillance system.

Test 1

After the training period the first test could be performed. The volunteer's learnt BN was used during this evaluation test. In addition, it was noted that the volunteer rarely entered into block 7 during the training period. Hence, block 7 was set as a restricted area block without notifying the volunteer. The intention was to demonstrate the detection of entry into a restricted area. Furthermore, the volunteer was not asked to perform any specific action during this test. A summary of the evaluation test results is presented below. The detailed results can be found in Appendix B, section B.4.1.

1. Blob Count

After comparing the prototype output to the ground truth data it was found that only one blob was present in all the frames. This meant that for this particular test the blob count accuracy was 100%.

2. Tracking of the Blob

Throughout the evaluation test sequence the same object was matched in each frame. This meant that for this particular test the tracking accuracy was 100%.

3. Action Recognition

The prototype surveillance system achieved an accuracy of 89% in action recognition. The main reason for this rather lower than expected action recognition accuracy is that at times the volunteer performed actions in line with the camera field of view instead of across the field of view of the camera. In addition, in a few frames the volunteer walked extremely slowly, as a result the action was detected as standing instead of walking. Thus, such actions were incorrectly recognised as standing and reducing the accuracy of the prototype surveillance system. This fact demonstrates one of the disadvantages of using a displacement metric for action recognition when the action performed in not across the field of view of the camera. However, this metric was the most suitable action recognition metric because the camera that was used could not detect the depth of the object in the environment.

4. Action Prediction

The action prediction accuracy of the prototype surveillance system was 69%. This value was significantly reduced by the action predictions made in block 6. For instance, block 6 predicted 4 possible actions with very close normalized percentages of: *pos6s* walking 21%, *pos10* walking 25%, pos5 walking 25% and pos7 walking 23%. The correct action in the next frame tended to be *pos6s* walking which was the 4th top prediction and was not included in the action prediction evaluation. Thus, the accuracy could have been increased if more than the top 3 predictions were considered in the evaluation procedure.

5. Anomaly Detection

The accuracy of the anomalies that were detected was 89%. More encouragingly the false negative rate was 0%, whilst the false positive rate was 12%. This meant that no anomalies were missed whilst only a few positive anomaly detections were incorrect. This is highlighted by the false alarm rate that was 85%. This value may seem very large, but one has too keep in

mind several actions were incorrectly recognized which lead to many of the false alarms. Importantly, the volunteer ran out of the environment at the end of the test and this was correctly detected by the system.

6. Anomaly Prediction

The accuracy of the anomalies that were predicted was 85%. Such a high prediction accuracy was achieved by considering all the anomaly predictions instead of only the top 3 predictions. However, this value was pushed up by correctly predicting that anomalies will not happen. The positive predictive value that was achieved was 0%. The main reason for this result is that the only anomaly that happened in the environment was running out of the environment at the end of the test. This was a totally new action that the system had never experienced before, thus it is expected that a prediction can not be made. The negative predictive value that was achieved was 98%. That meant that almost no anomalies were missed out by the prediction method. Finally, all these factors led to a false alarm rate of 100%. Nevertheless, such predictions can direct an operator's attention to potential anomalous behaviour and allow more time to react to such anomalous behaviour.

7. Restricted Block Entry Detection

The entry into restricted blocks was also detected with 100% accuracy.

8. Restricted Block Entry Prediction

The accuracy of the restricted block entry prediction was 52%. This value was much lower than expected and was mostly reduced by the fact that predictions of entry into restricted block 7 were made for the duration of a blob's stay in a neighbouring block, namely block 6. In other words, the predictions were made several frames before they happened and were assumed to be false positive predictions until the blob actually moved into the restricted block. The important thing to note is that the potential for entry into a restricted area was detected correctly. Hence, the false alarm rate was 66%. The positive predictive value that was achieved was 34%. Again this value was lower than expected because predictions were made a while before they happened. If the false predictions were only considered the first time they were made, the accuracy goes up to 90% whilst the positive predictive value becomes 87%. Finally, the negative predictive value was 97%.

Test 2

Test 2 required the volunteer to run into the environment, crouch in block 11 for some time then run out of the environment. The purpose of the test was to demonstrate the ability of the prototype surveillance system to detect and predict anomalous behaviour, because running across the environment and crouching in block 11 were rarely performed during the training period so they should be detected as anomalous actions. There were no restricted areas that were set up and the BN that was used was the one that was learnt using the volunteer's actions. A summary of the evaluation test results is presented below. The detailed results can be found in Appendix B, section B.4.2.

1. Blob Count

Similar to test 1, the blob count accuracy was 100%.

2. Tracking of the Blob

In addition, the tracking accuracy for this particular test was 100%.

3. Action Recognition

The prototype surveillance system achieved an accuracy of 99% in action recognition. The main reason for this high accuracy rate was that unlike in test 1 the actions performed were across the field of view of the camera.

4. Action Prediction

The action prediction accuracy of the prototype surveillance system was 85%. Like in test 1, this accuracy could have been increased if more than the top 3 predictions were considered in the evaluation procedure. In addition, this test showed that when a new action happens in the environment, the action is not predicted correctly the first time it happens, but the system learns the action and will predict it correctly the next time it happens.

5. Anomaly Detection

The accuracy of the anomalies that were detected was 46%. This percentage is obviously lower than what was expected, however it is very misleading. An analysis of the results revealed that the system initially detected the crouching action in block 11 correctly as an anomaly and then began to learn the action. Hence, it stopped detecting the action as an anomaly due to its learnt behaviour patterns. This fact shows the ability of the system to learn

patterns of activity and adapt to new patterns by incorporating them into its learnt patterns which is demonstrated in the next test.

Nevertheless, the accuracy statistic does not reveal that the system correctly recognised the running action across the environment and the initial crouching action in block 11. These were the goals of the test. The false negative rate of 74% highlights the fact that the system began to learn the crouching action in block 11. The false positive rate of 12% indicated that the anomaly detections that happened were mostly correct. The false alarm rate was only 21%.

Finally, it must be noted that when the system is in live mode, it still learns from the behaviours in the environment. This is the reason why the system adapted to the anomalous behaviour. If this is not the desired action that the system should take, then the system can be set up not to learn anomalous behaviour until the operator instructs it to.

6. Anomaly Prediction

The accuracy of the anomalies that were predicted was 41%. This value was also lower than expected because of the previously mentioned fact that there was not much activity in block 11 during the training period, hence the system began to adapt to the crouching action in block 11 by learning it as an accepted behaviour and thus stopped predicting it as an anomaly. On the other hand, the first time volunteer ran into the environment the anomaly was not predicted properly because the system had never encountered it before. Thereafter, the prediction of the anomalous running action was correct. This is to be expected because even in real life if one has not experienced an anomaly he or she will not predict the occurrence of such an anomaly.

More encouragingly, the positive predictive value was 82%. This meant that very few false anomaly predictions were made. However, owing to the previously mentioned reasons the negative predictive value was only 38%. The false alarm rate was only 18%. These results support the fact that the when an anomalous activity has never been experienced before it can not be predicted correctly.

Test 3

Test 3 was the last test that used the volunteer's trained BN. The purpose of this test was to demonstrate the ability of the prototype surveillance system to adapt to behaviour it had not experienced during the training process. This behaviour was observed in test 2, although the goal of the test was different resulting in different accuracy percentages. The new behaviour should initially be detected as anomalous behaviour and then the system should adapt to the behaviour and regard it as a regular behaviour. In addition, the ability of the system to detect different individuals will be tested. The volunteer was asked to run into the environment, bend down in block 8 and crawl forwards. This action was not done during the training period and it is hoped that the system will detect this action as an anomaly and then start to learn the action. The volunteer was then asked to exit the environment for more than 50 frames and then come back into the environment and stand in block 7. This was anomalous activity based on the training sequence. This was done in order to test whether the system can detect a new person entering the environment. In addition, block 6, which was the most common block in the training period, was set up as a restricted block. A summary of the evaluation test results is presented below. The detailed results can be found in Appendix B, section B.4.3.

1. Blob Count

The blob count accuracy was 100%.

2. Tracking of the Blob

The prototype video surveillance system correctly identified two different blobs in the environment. Hence, the tracking accuracy was 100%.

3. Action Recognition

The prototype surveillance system achieved an accuracy of 92% in action recognition. The few incorrect action recognitions were as a result of the volunteer crawling too slowly which resulted in the action being recognised as standing instead of walking. This can always be corrected by adjusting the Euclidean distance range for each action.

4. Action Prediction

The action prediction accuracy of the prototype surveillance system was 71%. This accuracy was slightly lowered by the new crawling action that had not been experienced before; hence,

it could not be predicted properly. In addition, several actions that were properly predicted did not fall into the top 3 predictions, thus contributing to the lower accuracy. A possible solution to this problem would be to consider all action that are predicted with a certain percentage threshold in determining the accuracy of the action predictions instead of using only the top 3 predictions.

5. Anomaly Detection

The accuracy of the anomalies that were detected was 86%. This accuracy was slightly lowered by some actions being incorrectly recognized. This is highlighted by the false negative rate of 25% and a false positive rate of 7%. The false alarm rate was 16%. The most important fact about this evaluation test is that the system adapted to the new anomalous actions that were being performed, i.e. crawling in block 8 and standing in block 7, and after a while these actions were no longer regarded as anomalies. The time to adapt to the new actions depends on the amount of training data that each block BN has. For instance, the BN in block 8 did not have a lot of training data so it adapted quickly to the new actions, whereas the standing action in block 7 took longer to adapt to.

6. Anomaly Prediction

The accuracy of the anomalies that were predicted was 71%. The positive predictive value that was achieved was 57%. The negative predictive value that was achieved was 80%. The main reason for a lower negative predictive value was that the new anomalous actions could not be predicted properly until the system experienced them at least once. The false alarm rate was 43%. Although the false alarm rate is a bit high, such predictions can direct an operator's attention to potential anomalous behaviour and allow more time to react to such anomalous behaviour.

7. Restricted Block Entry Detection

The entry into restricted blocks was also detected with 100% accuracy.

8. Restricted Block Entry Prediction

The accuracy of the restricted block entry prediction was 71%. This value was slightly reduced by the fact that predictions of entry into restricted block 6 were made for the duration of a blob's stay in the neighbouring blocks, namely block 5 and block 7. In other words, the predictions were made several frames before they happened and were assumed to be false positive predictions until the blob actually moved into the restricted block. Hence, the false

alarm rate was 59%. The positive predictive value that was achieved was 41%. Finally, the negative predictive value was 100%.

Test 4

Test 4 was the first test that used the BN that was trained using common patterns in the Agent's Lab over a prolonged period of time. The purpose of this test was to see if the prototype system could differentiate some of the actions of the volunteer from the common actions in the environment learnt over a 3 month period. In addition, blocks 4, 7 and 11 were labelled as restricted areas. A summary of the evaluation test results is presented below. The detailed results can be found in Appendix B, section B.4.4.

1. Blob Count

The blob count accuracy was again 100%.

2. Tracking of the Blob

The prototype video surveillance system correctly identified one individual in the environment with a tracking accuracy of 100%.

3. Action Recognition

The prototype surveillance system achieved an accuracy of 99% in action recognition.

4. Action Prediction

The action prediction accuracy of the prototype surveillance system was 81%.

5. Anomaly Detection

The accuracy of the anomalies that were detected was 99%. The false negative rate was 0%, whilst the false positive rate was 1%. The false alarm rate was only 3%. The most important fact about this evaluation test is that the system detected a crouching action from block 5 into block 9 which was a true anomalous action.

6. Anomaly Prediction

The accuracy of the anomalies that were predicted was 89%. This value was lowered mainly by the fact that the block 5 BN did not predict that the blob would remain in block 5 in the next frame as the top three predictions. Encouragingly, the positive predictive value that was achieved was 58%. The negative predictive value that was achieved was 96%. The false alarm rate was only 42% which meant that several of the anomalies were predicted correctly.

7. Restricted Block Entry Detection

The entry into restricted blocks was detected with 100% accuracy.

8. Restricted Block Entry Prediction

The accuracy of the restricted block entry prediction was 61%. This value was slightly reduced by the fact that predictions of entry into restricted blocks were made for the duration of a blob's stay in a neighbouring block. Instead of viewing these predictions as false prediction, they should be viewed as potential predictions which can serve as warning indicators for operators. The false alarm rate was 38%. The positive predictive value that was achieved was 47%. Finally, the negative predictive value was 100%.

Test 5

The last evaluation test that was performed was designed to test the prototypes ability to detect a re-entry into the environment by the volunteer after a short time period. Similar to test 4, the BN that was trained using common patterns in the Agent's Lab was used. The purpose of this test was to see if the prototype system could differentiate some of the actions of the volunteer from the common actions in the environment learnt over a 3 month period. In addition, block 8 was set up as a restricted block. A summary of the evaluation test results is presented below. The detailed results can be found in Appendix B, section B.4.5.

1. Blob Count

The blob count accuracy was again 100%.

2. Tracking of the Blob

The prototype video surveillance system correctly identified the same blob re-entering the environment with a tracking accuracy of 100%.

3. Action Recognition

The prototype surveillance system achieved an accuracy of 99% in action recognition.

4. Action Prediction

The action prediction accuracy of the prototype surveillance system was 96%.

5. Anomaly Detection

The accuracy of the anomalies that were detected was 99%. The false negative rate was 11%, whilst the false positive rate was only 1%. The false alarm rate was only 11%. The most important fact about this evaluation test is that the system detected the volunteer running back into the environment.

6. Anomaly Prediction

The accuracy of the anomalies that were predicted was 90%. The positive predictive value that was achieved was 35%. The negative predictive value that was achieved was 99%. However, the false alarm rate was 65% which meant that several of the anomalies that were predicted did not happen. Again, these predictions should be viewed as potential anomalies.

7. Restricted Block Entry Detection

The volunteer did not enter into the restricted block and this was reflected by the prototype system with an accuracy of 100%.

8. Restricted Block Entry Prediction

Similarly, there were no predictions made by the system with regards to entering a restricted block, so the accuracy was 100%.

5.3.4 Summary of Robustness and Accuracy Evaluation Tests

When looking at the accuracy that the prototype surveillance system achieved several things should be kept in mind. Firstly, the environment was an experimental environment where environmental factors were controlled. Secondly, the evaluations were based on the top 3 predictions instead of a single maximum prediction. This fact allowed high accuracy results. Even higher accuracies can be achieved by considering all predictions above a certain percentage threshold. Finally, the evaluation tests that were performed were designed to test specific aspects of the prototype system.

The average blob counting and tracking accuracy of the prototype system was 100%. This was mostly achieved because only one volunteer was involved in the evaluation tests, which is what the prototype system was designed to handle. The average action recognition accuracy was 96%. This value could have been made higher by tweaking the Euclidean distance range for each action to suit the volunteer. The average action prediction accuracy

was 82%. This is a very high accuracy for predictions and was only possible by observing a set of probable actions instead of one definite action. If a maximum value was observed this accuracy would be much lower.

A common trend appeared throughout the evaluation tests. Most of the time when a future action is incorrectly predicted it is a result of the BN not observing that action before, i.e. it is a new action that is happening. This usually resulted in an anomalous behaviour being detected in the next frame or until the BN learns this action pattern and stops detecting it as an anomaly. One has to note that learning the patterns of anomalous behaviours is not always the intended procedure; hence, the system can be modified to only learn anomalous behaviour patterns only when an operator instructs it to.

The average anomaly detection accuracy was 83%. Ideally this value should have been 100%. However, the accuracy was lowered by the incorrect recognition of some actions which resulted in them being incorrectly detected as anomalies. In addition, test 2 significantly reduced the average accuracy. The average false negative rate was 22%, which meant that only a few anomalies were missed by the system. The average false positive rate was also 7%. Finally, the average false alarm rate was 27%. This value was mostly bumped up by test 1 in which several actions where incorrectly detected as anomalies due to incorrect action recognition. Most importantly, the actual anomalous activity that happened during the evaluation tests was detected correctly.

The average anomaly prediction accuracy was 75%. This value was significantly bumped up by the fact that the non occurrence of anomalies was mostly correctly predicted, as is indicated by the negative prediction rate average of 82%. The average positive prediction rate of 46% meant that a few false positive predictions were made. This is also highlighted by the average false alarm rate of 57%. Evaluation test 1 contributed the most to this average false alarm rate and also reduced the average accuracy of the predictions. Instead of looking at this false alarm rate negatively, it shows the ability of the system to predict potential anomalous activity even though the activity may not have happened in the next frame.

The average restricted block entry detection was 100%. This accuracy was expected because detecting entry into a restricted area is a trivial task. The average restricted block prediction accuracy was 71%. Again, this value was bumped up by correctly predicting a blob will not

enter into a restricted area which is shown by the average negative prediction percentage of 99%. The average positive prediction percentage is 41%. In addition, the false alarm rate average was 54%. These predictions show potential of an individual to enter into a restricted blob.

Another important fact was that in evaluation test 2 and 3, the prototype system demonstrated its ability to detect new activities as being anomalous activities. After a while the prototype system adapted to these new behaviours and did not regard it as anomalies thereafter. Finally, the evaluation tests demonstrated the use of an individually trained BN collection as well as a commonly learnt BN collection. The predictions made by the BN that learnt common patterns were much higher than the BN that learnt individual patterns.

5.3.5 Evaluation using PETS 2002 Datasets

Although the prototype video surveillance system was designed for the Agents Research Lab an evaluation of its robustness and accuracy using the PETS 2002 dataset [41] would allow comparison with other researchers' results. Slight modifications had to be made to the prototype video surveillance system because several of the assumptions made about the Agents Lab would not hold for the PETS dataset.

In particular, the filtering metric for the size of the foreground objects had to be reduced to capture most of the individuals in the datasets. In addition, the small blob merging Euclidean distance threshold had to be reduced to minimize incorrect blob segmentation. The maximum tracking Euclidean distance was reduced from 150 pixels to 50 pixels because there was no running action in the dataset. Finally, the Euclidean distance range for standing was reduced because the individuals in the image sequences walk very slowly. Only the PETS 2002 [41] dataset 1 was used in this evaluation because the length of dataset 2 and 3 [41], 1471 [41] and 1295 [41] and respectively, would make extracting ground truth data an extremely laborious task. During the training sequence the most common action was walking.

Dataset 1 Results

A new BN collection was set up for the PETS 2002 [41] dataset 1 and was trained using the training image sequence, 377 frames, that was provided. Following that, the testing sequence, 653 frames, was used to evaluate the accuracy and robustness of the prototype video

surveillance system. Ground truth data was extracted manually from the PETS 2002 [41] testing dataset and was compared to the prototype surveillance system results and reference images. A snap shot of the prototype surveillance system working on the PETS 2002 dataset 1 is shown below in Figure 41.

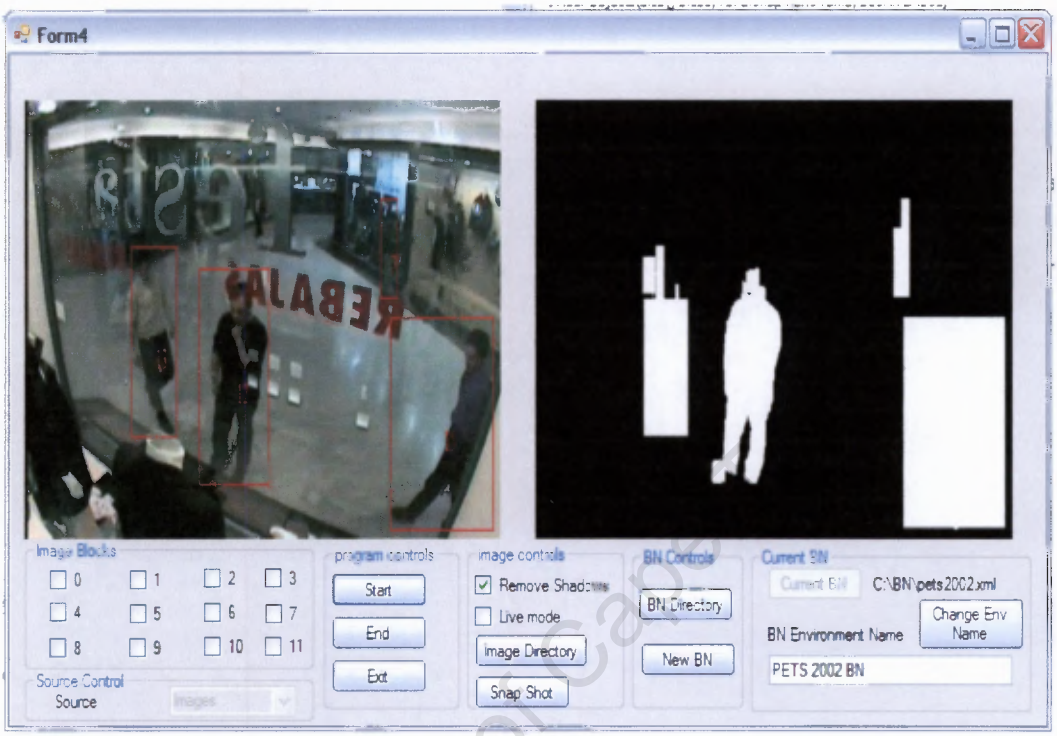


Figure 41: Prototype system working on Pets 2002 dataset 1

A summary of the results that were obtained is presented below. The detailed results can be found in Appendix B, Section B.5.

1. Blob Count

The blob count accuracy was 64%. This blob count percentage is very encouraging, because the PETS 2002 [41] dataset 1 has many challenging scenes that the prototype surveillance was not designed for. These include, object occlusion (e.g. Frame 420), group of people walking together (e.g. Frame 357), individuals standing close together (e.g. Frame 623), reflections in the window (e.g. Frame 10) and illumination variance which was also noted by Ellis [40].

This accuracy was mainly reduced by the fact that the individuals in the top of the environment were very small in size and the clothes of the individuals were the same as the background (camouflage). Hence, these individuals filtered out as larger noise regions, e.g. Frame 200 and Frame 550. However, one must note that these individuals were correctly tracked when they were picked up by the prototype system.

In addition, the person walking into the top of the environment from the left hand side is blocked by the writing on the shop window, e.g. Frame 87. Hence, the individual is very hard to spot even with the naked eye. These facts are highlighted by the false negative rate of 37%. In addition, there were a few false blob counts due to the reflection in the window and illumination variance. However, the occurrence of the false positive counts, or false alarm rate, was very low at 0.5% with a false positive rate was 7%.

Finally, the prototype system picked up every individual in the environment at some point in time. When a group of people was split into two individuals, the prototype system assigned new blob numbers to each of the individuals. Likewise when two or more individuals joined to form a group the group was assigned a new blob count, e.g. Frame 440 and Frame 648. Thus, the final blob count for the prototype system was 18. By naked eye, the total number of individuals in the environment was 7. This was also reflected by the prototype system, although the final blob number is wrong due to the above mentioned reasons. In comparison, Cucchiara [8], reports that his system only detected 5 individuals in dataset 1. In addition, Cucchiara [8] reports a false negative count of 115. The true positive count is not provided.

2. Tracking of the Blobs

The prototype video surveillance system had a tracking accuracy of 99% for the individuals that were correctly identified in the scene. When the individual was missed no tracking data was recorded.

3. Action Recognition

The prototype surveillance system achieved an accuracy of 96% in action recognition for the individuals that it tracked. This was mainly possible because most of the actions were across the field of view of the camera. In addition, the action recognition Euclidean distance ranges were tweaked to cater for the slow moving individuals in the scene. Importantly there was no running action in the environment and this was reflected by the prototype system.

4. Action Prediction

Likewise, the action prediction accuracy of the prototype surveillance system was 96%. This high accuracy was only possible by considering a set of the top 3 predictions instead of a single maximum prediction. In addition, the accuracy could have been even higher if all prediction above a certain threshold were considered.

5. Anomaly Detection

The accuracy of the anomalies that were detected was 96%. This accuracy was bumped up by the fact that the system correctly detected normal behaviour as not being an anomaly. However, this accuracy doe not reveal the fact the prototype system began to learn the anomalous behaviours such as the individuals who stopped in front of the shop window, e.g. Frame 475 and Frame 583. These actions were not performed during the training period and were correctly identified as anomalies. After a while the prototype system learnt these behaviours and stopped detecting them as anomalies. In addition the individual in the top of the environment slows down to look at the people across him which was an anomalous behaviour based on the training data and was correctly identified by the prototype system. The false negative rate was only 7%, whilst the false positive rate was only 4%. In addition, the false alarm rate was only 26%.

6. Anomaly Prediction

The accuracy of the anomalies that were predicted was 92%. Again, this value was bumped up by the fact that the system correctly predicted that anomalies would not occur. This is shown by the negative predictive value that of 99%. The positive predictive value that was achieved was an encouraging 57%. However, the false alarm rate was 43% which meant that several of the anomalies that were predicted did not happen. Again, these predictions should be viewed as potential anomalous activity predictions.

5.4 Conclusion

This Chapter presented the tests that were performed during the designing phase of the prototype system and the evaluation period, including the results that were obtained. The first part of the Chapter discussed the evaluation tests that were needed to make certain design time decisions with regards to both the IU and AI components. These included determining the merging Euclidean distance for small blobs that belong to the same individual blob,

determining the size of the larger noise regions, determining the Euclidean distance to track and match individuals in the environment and finally determining the Euclidean distance range for the recognition of actions in the environment.

The second part of the Chapter discussed the evaluation tests that were performed to test the robustness and accuracy of the prototype video surveillance system. The first test that was performed was the inference value test to make sure that the BN inference was correct. The inference values calculated by BaBe [62] were identical to those calculated by JavaBayes [57]. Following that, an evaluation of the overall prototype system was performed using a volunteer to perform actions in the environment. Two different BN collections were used for these tests to see if there was any difference in the BN learnt using the volunteer's training data and the BN learnt with actions in the environment over a three month period. The average blob count and tracking accuracy was 100%. The average action recognition accuracy was 96%. The average action prediction accuracy was 82%. The average anomaly detection accuracy was 83%. The average anomaly prediction accuracy was 75%. The average entry into restricted block detection was 100%. Finally, the average restricted block entry prediction was 71%.

The Chapter concluded with a discussion of the prototype video surveillance system performance on the PETS 2002 [41] dataset 1. The next Chapter concludes the dissertation.

Chapter 6

Conclusion and Future Work

This Chapter concludes the dissertation by summarising the research conducted and presenting the main prototype evaluation results. In addition, the Chapter presents possible directions for future work which can be done to extend and build upon the ideas presented in this dissertation.

6.1 Summary of the Research Done

The focus of this research was to detect and predict anomalous behaviour of individuals in an environment using a camera as a way to capture the information about the environment. The monitored environment that was chosen was the UCT Agents Research Lab. A prototype video surveillance system was developed in order to demonstrate the research conducted during this dissertation and also to achieve the research goals that were set out. The prototype that was developed was specifically developed for the environment and the environmental factors that were experienced. However, the ideas presented in the dissertation are not specific to the chosen environment and it is hoped that they will be tested out by other researchers in various real world environments.

The prototype video surveillance system comprised of an IU and AI component. The IU component was developed in order to provide data capturing and information gathering capabilities to the AI component by processing image sequences that were received from a video camera into environment data. This IU component was modified so that it could use saved images as a source in order to test the prototype system on the PETS 2002 [41] dataset.

This processed data was then passed onto the AI component so that it could be analysed in order to learn an individual's action and behaviour patterns. These learnt action patterns were used to make predictions about future actions. In addition the learnt patterns were used to detect and predict anomalous actions in the environment which did not follow common action trends. A collection of different BNs was used at the heart of the AI component to perform all knowledge representation, modelling and probabilistic reasoning.

The actions that were chosen were specific to the environment that was being modelled. In a real world environment, several other actions would have to be incorporated into the system to make it more realistic. In addition, the assumptions made on the transition of a person from one action to another were specific to the selected environment. However, most of the assumptions will hold in real world scenarios, e.g. if a person is walking in the current frame he could walk in the next frame or could decide to stand still.

The prototype video surveillance system was evaluated by using a volunteer who performed certain action in the environment to test specific aspects of the prototype system. Two different BN collections were used in the evaluation tests. The first was the BN collection which learnt the volunteer's actions patterns and the second was the BN collection that learnt the common patterns in the environment over a three month period. In addition, the prototype system was evaluated using a real world dataset, namely the PETS 2002 [41] dataset 1 with promising results. This fact showed that the system could be used on larger datasets that is was not designed for. Hence, it is believed that with minor modifications the prototype system would be able to scale to real world environments.

6.2 Achievement of Research Goals

Various techniques were used to achieve the research goals that were set out. A summary of these techniques is presented below:

1. Can a Bayesian Network be used to model the environment and peoples' behaviour in that environment?

   The first research question was answered by analysing the chosen environment in order to determine the most appropriate BN structure for the environment and also to determine what kind of actions the BN would be modelling. A thorough positive evaluation of the prototype surveillance system provided good supporting evidence for the use of a BN to model people's behaviour in an environment. This was highlighted by the robustness and accuracy test 3 during which the prototype system learnt and adapted to a new behaviour that was introduced. In addition, the system performed very well on the PETS 2002 [41] dataset 1 which it was not developed for.

2. How can the massive amount of data needed to model an environment be represented so that a Bayesian Network can be used to model the environment without sacrificing vital information about the environment?

   By dividing the environment into a grid of blocks the amount of data needed to model the environment was drastically reduced without the loss of vital environment information. In addition, by using a collection of BNs instead of a single BN to model the environment the bi-directional causal relationships between the neighbouring blocks in the environment grid were dealt with and the data was reduced even further whilst maintaining the vital causal relationships of the environmental grid.

3. Can anomalous behaviour be detected from learnt behavioural patterns?

   This research question was answered by specifying anomalies as actions that did not conform to the most common action patterns in each grid block. This was demonstrated by the fact that the prototype surveillance system picked up the actions that were not common activities in the environment as anomalous behaviours during the evaluation tests. In addition, the prototype system showed its ability to learn these new anomalous activities and integrate them into the common action patterns. However, if this is not the desired system behaviour that is needed then the system can be set up so that it does not incorporate the anomalous behaviour into its common patterns.

4. Can predictions of human behaviour be done in near real time?

   Near real-time performance was achieved by the prototype surveillance system on the computer that the prototype was designed on at 7 fps with only a slight time lag between the pictures being displayed and the actual actions happening in the environment. In addition, real-time performance of the 7 fps was achieved on a faster computer. This was accomplished by using a collection of small BNs that reduced the processing time of all query operations.

6.3 Summary of Results

After performing the evaluation tests in the Agent's Lab, the average blob counting and tracking accuracy of the prototype system was 100%. The average action recognition

accuracy was 96%. By considering a set of action probabilities instead of a singular maximum prediction, the average action prediction accuracy was 82%. The average anomaly detection accuracy was 83% with an average false negative rate of 22% and average false positive rate of 7%. This meant that the anomalies that did occur were correctly detected and is highlighted by the low average false alarm rate of only 27%. This value was mostly bumped up by test 1 in which several actions where incorrectly detected as anomalies.

The average anomaly prediction accuracy was 75%. This value was significantly bumped up by the fact that the non occurrence of anomalies was mostly correctly predicted, as is indicated by the negative prediction rate average of 82%. However, the average positive prediction rate of 46% meant that a few false positive predictions were made. This is also highlighted by the average false alarm rate of 57%. Instead of looking at this false alarm rate negatively, it shows the ability of the system to predict potential anomalous activity even though the activity may not have happened in the next frame.

The average restricted block entry detection was 100%. The average restricted block entry prediction accuracy was 71%. Again, this value was bumped up by correctly predicting a blob will not enter into a restricted area which is shown by the average negative prediction percentage of 99%. The average positive prediction percentage was 40% whilst the false alarm rate was 54%. These predictions also show potential of an individual to enter into a restricted blob.

Another important fact was that in evaluation test 2 and 3, the prototype system demonstrated its ability to detect new activities as being anomalous activities. Finally, the evaluation tests demonstrated the use of an individually trained BN collection as well as a commonly learnt BN collection. The accuracy of the predictions made by the BN that learnt common patterns were much higher than the BN that learnt individual patterns.

The prototype system was then evaluated using the PETS 2002 [41] dataset 1. Although it was not developed for this dataset the results were very encouraging. The blob count accuracy was 64%, mainly due to the fact that some of the individuals in the environment were small in size, resulting in them being filtered out as larger noise regions. More encouragingly, the prototype system picked up all 7 individuals that were noted using the

naked eye. Groups of individuals were correctly split up into two separate individuals and merging of individuals was correctly recognized as being a group.

For the individuals that were correctly recognised the tracking accuracy was 99%, the action recognition accuracy was 96%, the action prediction accuracy was 96%, the anomaly detection accuracy was 96% and finally the anomaly prediction accuracy was 92%. These high accuracy percentages were aided by the fact that the actions were done across the field of view of the camera and a set of the top 3 predictions were used in predicting future actions and anomalies.

6.4 Contributions of the Research

The main contribution of this research is in the field of Artificial Intelligence, specifically:

1. Provides further support to the combined use of IU and AI, specifically the use of a BN, in video surveillance systems;
2. The use of a BN to model an environment and the behaviours that are happening in that environment;
3. Provides a unique solution to modelling an environment that is split into a grid of blocks by using a combination of different BNs to model it;
4. Adds to the existing research on learning data patterns using BNs, i.e. data mining, and the existing research on predicting human behaviours based on those learnt patterns;
5. Provides further support for detecting and predicting anomalous human behaviour based on the learnt patterns and heuristic knowledge;
6. Finally, and most importantly, the research contributes to the body of knowledge in surveillance systems by providing support for the use of BNs in surveillance systems;

6.5 Future Work

The research conducted during this dissertation established a proof of concept about the use of AI techniques, in particular a BN, to detect and predict anomalous actions in a prototype video surveillance system. Hence several directions are possible as future work. Some of these include:

- Applying the techniques used in this research to a less experimental environment such as an outdoor environment which has more realistic environmental factors;
- Application of the research ideas and techniques to an environment consisting of several cameras and/or sensors to get a larger environment to monitor;
- Extending the set of actions that were monitored by the prototype video surveillance system to get a more realistic model of the environment. E.g. adding complex actions such as interactions between individuals in the environment or leaving and picking up object in the environment;
- Adding several other objects into the environment;
- Combining the images of each blob to get an image sequence that can be used as evidence in the case of security surveillance systems;
- An investigation into using more than one time step in action prediction;
- An investigation into modelling group dynamics instead of individuals.

6.5 Concluding Remarks

The research that was conducted during this dissertation provides further support for the use AI techniques, such as a BN, in video surveillance systems. It is hoped that in the future more researchers will use the ideas presented during the dissertation. In addition, it is hoped that researchers will expand and build on the ideas that were presented.

References

1. Friedman, N. and Russell, S. 1997. Image segmentation in video sequences: a probabilistic approach. In *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, (Brown University, Providence, Rhode Island, USA, August 1-3, 1997). Morgan Kaufmann, San Francisco, CA, 175 - 181.
2. Bodor, R., Bennett, J. and Papanikolopoulos, N. 2006. Vision-Based Human Tracking and Activity Recognition. Retrieved April 4, 2006, from Artifical Intelligence, Robotics and Vision Laboratory, Department of Computer Science and Engineering, University of Minnesota: http://mha.cs.umn.edu/Papers/Vision_Tracking_Recognition.pdf
3. Wei, G. and Sethi, I. K. 2000. Omni-face detection for video/image content description. In *Proceedings of the 2000 ACM Multimedia Workshops*, (Los Angeles, California, United States, 2000). ACM Press, New York, NY, USA, 185-189.
4. Stauffer, C. and Grimson, W. 2000. Adaptive background mixture models for real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, (2000). IEEE Computer Society, Los Alamitos, CA, 747-757.
5. Hu, W., Tan, T., Wang, L. and Maybank, S. 2004. A survey on visual surveillance of object motion and behaviours. *IEEE transactions on systems, man, and cybernetics—part c: applications and reviews,* vol. 34, no. 3, (August 2004). IEEE Systems, Man, and Cybernetics Society, 334-352.
6. Sidenbladh, H. 2004. Detecting Human Motion with Support Vector Machines. In *17th International Conference on Pattern Recognition (ICPR'04)*, 2, (2004). IEEE Computer Society, Los Alamitos, CA, 188-191
7. Lipton, A. J., Fujiyoshi, H. and Patil, R. S. 1998. Moving target classification and tracking from real-time video. In *Proc. IEEE Workshop on Applications of Computer Vision*, (Princeton, NJ, USA, October 1998). IEEE Computer Society, Los Alamitos, CA, 8–14.
8. Cucchiara, R., Grana, C., and Tardini, G. 2004. Track-based and object-based occlusion for people tracking refinement in indoor surveillance. In *Proceedings of the ACM 2nd international Workshop on Video Surveillance and Sensor Networks* (New York, NY, USA, October 15, 2004). VSSN '04. ACM Press, New York, NY, 81-87.

9. Wang, G., Wong, T. and Heng, P. 2005. Real-time surveillance video display with salience. In *Proceedings of the third ACM international workshop on Video surveillance and sensor networks*, (Hilton, Singapore, 2005). ACM Press New York, NY, USA, 37-44.
10. Ivanov, Y., Stauffer, C., Bobick, A. and Grimson W. L. E. 1999. Video Surveillance of Interactions. In *Proceedings of the Second IEEE Workshop on Visual Surveillance*, (1999). IEEE Computer Society, Los Alamitos, CA, USA, 82.
11. Lou, J., Liu, Q., Tan, T., and Hu, W. 2002. Semantic Interpretation of Object Activities in a Surveillance System. In *Proceedings of the 16th international Conference on Pattern Recognition (Icpr'02) Volume 3*, vol. 3, (August 11 - 15, 2002). IEEE Computer Society, Los Alamitos, CA, USA, 30777.
12. Baker, C. L., Tenenbaum, J. B. and Saxe, R. R. (2006). Bayesian models of human action understanding. In *Advances in Neural Information Processing Systems 18*.
13. Bezzi, M. and Groenevelt, R. 2006. Towards understanding and modeling individual behavior and group dynamics. In *Proceedings of the Pervasive 2006 Workshop "Pervasive Technology Applied - Real-World Experiences with RFID and Sensor Networks"*, (Dublin, Ireland, May 7, 2006).
14. Thirde, D., Borg, M., Ferryman, J., Fusier, F., Valentin, V., Bremond, F., and Thonnat, M. 2006. A Real-Time Scene Understanding System for Airport Apron Monitoring. In Proceedings *of the Fourth IEEE international Conference on Computer Vision Systems* (January 04 - 07, 2006). ICVS. IEEE Computer Society, Los Alamitos, CA, 26.
15. Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. P. 1997. Pfinder: Real-Time Tracking of the Human Body. *IEEE Trans. Pattern Anal. Mach. Intell.* vol. 19, no. 7, (July, 1997). IEEE Computer Society, Los Alamitos, CA, 780-785.
16. Nascimento, J. C., Figueiredo, M. A. T. and Marques, J. S. 2005. Segmentation and Classification of Human Activities. In *Proceedings of International Workshop on Human Activity Recognition and Modelling (HAREM 2005 - in conjunction with BMVC 2005)* (Oxford, UK, September 2005), 79-86.
17. Cohen, I., Sebe, N., Cozman, F. G., and Huang, T. S. 2003. Semi-supervised learning for facial expression recognition. In *Proceedings of the 5th ACM SIGMM international Workshop on Multimedia Information Retrieval* (Berkeley, California, November 07 - 07, 2003). MIR '03. ACM Press, New York, NY, 17-22.
18. Park, S. and Aggarwal, J. K. 2003. Recognition of two-person interactions using a hierarchical Bayesian network. In *First ACM SIGMM international Workshop on Video*

Surveillance (Berkeley, California, November 02 - 08, 2003). IWVS '03. ACM Press, New York, NY, 65-76.

19. Park, S. and Trivedi, M. M. 2006. Analysis and query of person-vehicle interactions in homography domain. In *Proceedings of the 4th ACM international Workshop on Video Surveillance and Sensor Networks* (Santa Barbara, California, USA, October 27 - 27, 2006). VSSN '06. ACM Press, New York, NY, 101-110.
20. Jacobs, N. and Pless, R. 2006. Real-time constant memory visual summaries for surveillance. In *Proceedings of the 4th ACM international Workshop on Video Surveillance and Sensor Networks* (Santa Barbara, California, USA, October 27 - 27, 2006). VSSN '06. ACM Press, New York, NY, 155-160.
21. Chen, D., Yang, J., and Wactlar, H. D. 2004. Towards automatic analysis of social interaction patterns in a nursing home environment from video. In *Proceedings of the 6th ACM SIGMM international Workshop on Multimedia information Retrieval* (New York, NY, USA, October 15 - 16, 2004). MIR '04. ACM Press, New York, NY, 283-290.
22. Bashir, F., Porikli, F. 2006. Performance Evaluation of Object Detection and Tracking Systems. In *Proceeding of the 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (New York, USA, June 18, 2006). PETS 2006. IEEE Computer Society, Los Alamitos, CA, 7-14.
23. Ribeiro, P. C., and Santos-Victor, J. 2005. Human Activity Recognition from Video: modeling, feature selection and classification architecture. In *Proceedings of International Workshop on Human Activity Recognition and Modelling (HAREM 2005 - in conjunction with BMVC 2005)* (Oxford, UK, September, 2005), 61-70.
24. Stauffer, C. and Grimson, W. 1999. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, (Fort Collins, CO, USA, June 23-25, 1999), IEEE Computer Society, Los Alamitos, CA, 246–252.
25. Russ, J. C. 1995. *The Image Processing Handbook (2nd Ed.)*. CRC Press, Inc. Boca Raton, Fla.
26. Normal Distribution. Retrieved July 1, 2007, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Normal_distribution
27. Mahalanobis distance. Retrieved July 1, 2007, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Mahalanobis_distance
28. Keller, G. and Warrack, B. (2000). *Statistics for Management and Economics (5th edition)* Duxbury, Pacific Grove, CA.

29. Koller, D., Weber, J., Huang, T. Malik, J. Ogasawara, G., Rao, B. and Russell, S. 1994. Toward robust automatic trafic scene analysis in real-time. In *Proceedings of the 12th Int'l Conference on Pattern Recognition (ICPR-94)*, (Jerusalem, Israel, October 9-13, 1994), 126-131.

30. Harville, M., Gordon, G., and Woodfill, J. 2001. Foreground Segmentation Using Adaptive Mixture Models in Color and Depth. In *Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video*, (Vancouver, BC, Canada, July 8 2001), IEEE Computer Society, Los Alamitos, CA, 3-11.

31. Finlayson, G. D., Hordley, S. D., and Drew, M. S. 2002. Removing Shadows from Images. In *Proceedings of the 7th European Conference on Computer Vision-Part IV* (May 28 - 31, 2002). A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Lecture Notes In Computer Science, vol. 2353. Springer-Verlag, London, 823-836.

32. Baba, M., Mukunoki, M., and Asada, N. 2004. Shadow removal from a real image based on shadow density. In *ACM SIGGRAPH 2004 Posters* (Los Angeles, California, August 08 - 12, 2004). R. Barzel, Ed. SIGGRAPH '04. ACM Press, New York, NY, 60.

33. Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. 2003. Detecting Moving Objects, Ghosts, and Shadows in Video Streams. *IEEE transactions on pattern analysis and machine intelligence, vol. 25, no. 10*, (OCTOBER 2003). IEEE Computer Society, Los Alamitos, CA, 1337-1342

34. Welch, G. and Bishop, G. 1995 *An Introduction to the Kalman Filter*. Technical Report. UMI Order Number: TR95-041., University of North Carolina at Chapel Hill.

35. Haritaoglu, I., Harwood, D., and David, L. S. 2000. W4: Real-Time Surveillance of People and Their Activities. *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 22, no. 8 (Aug. 2000), 809-830.

36. Tansuriyavong, S. and Hanaki, S. 2001. Privacy protection by concealing persons in circumstantial video image. In *Proceedings of the 2001 Workshop on Perceptive User interfaces* (Orlando, Florida, November 15 - 16, 2001). PUI '01, vol. 15. ACM Press, New York, NY, 1-4.

37. Senior, A. 2002. Tracking People with Probabilistic Appearance Models. In *Proceeding of the 3rd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (Copenhagen, Denmark, June 1, 2002). PETS 2002. IEEE Computer Society, Los Alamitos, CA, 48-55.

38. Siebel N. T. and Maybank, S. J. 2001. Real-time tracking of pedestrians and vehicles. In *Proceeding of the 2nd IEEE International Workshop on Performance Evaluation of*

Tracking and Surveillance (Kauai, Hawaii, USA, December 9, 2001). PETS 2001. IEEE Computer Society, Los Alamitos, CA.

39. Seibert, M., Rhodes, B. J., Bomberger, N., A., Beane, A., O., Sroka, J. J., Kogel, W., Kreamer, W., Stauffer, C., Kirschner, L., Chalom, E., Bosse, M. and Tillson, R. 2006. SeeCoast port surveillance In *Proceedings of SPIE Vol. 6204: Photonics for Port and Harbor Security II* (Orlando, FL, USA, April 18-19, 2006).
40. Ellis, T. 2002. Performance Metrics and Methods for Tracking in Surveillance. In *Proceeding of the 3rd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (Copenhagen, Denmark, June 1, 2002). PETS 2002. IEEE Computer Society, Los Alamitos, CA, 26-31.
41. PETS: Performance Evaluation of Tracking and Surveillance. 2005. Retrieved February 12, 2007, From Computational Vision Group, School of Systems Engineering, University of Reading: http://www.cvg.rdg.ac.uk/slides/pets.html
42. Lazarevic-McManus, N., Renno, J., Makris, D. and Jones, G. A. 2006. Designing Evaluation Methodologies: The Case of Motion Detection. In *Proceeding of the 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (New York, USA, June 18, 2006). PETS 2006. IEEE Computer Society, Los Alamitos, CA, 23-30.
43. Grabner, H., Roth, P. M., Grabner, M. and Bischof, H. 2006. Autonomous Learning of a Robust Background Model for Change Detection. In *Proceeding of the 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (New York, USA, June 18, 2006). PETS 2006. IEEE Computer Society, Los Alamitos, CA, 39-46.
44. Krahnstoever, N., Tu, P., Sebastian, T., Perera, A. and Collins, R. 2006. Multi-View Detection and Tracking of Travelers and Luggage in Mass Transit Environments. In *Proceeding of the 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (New York, USA, June 18, 2006). PETS 2006. IEEE Computer Society, Los Alamitos, CA, 67-74.
45. Smith, K., P. Quelhas, P. and Gatica-Perez, D. 2006. Detecting Abandoned Luggage Items in a Public Space. In *Proceeding of the 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (New York, USA, June 18, 2006). PETS 2006. IEEE Computer Society, Los Alamitos, CA, 75-82.
46. Lv, F., Song, X., Wu, B., Singh, V. K. and Nevatia, R. 2006. Left Luggage Detection using Bayesian Inference. In *Proceeding of the 9th IEEE International Workshop on*

Performance Evaluation of Tracking and Surveillance (New York, USA, June 18, 2006). PETS 2006. IEEE Computer Society, Los Alamitos, CA, 83-90.

47. Gonzalez, R. C. and Woods, R. E. 2002. *Digital Image Processing (2nd Edition).* Prentice Hall, Upper Saddle River, N.J.
48. Russel, S. J. and Norvig, P. 2003. *Digital Artificial Intelligence A Modern Approach (2 ed).* Prentice Hall, Upper Saddle River, N.J.
49. Duque, D., Santos, H. and Cortez, P. 2007. Prediction of Abnormal Behaviors for Intelligent Video Surveillance Systems. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining* (Honolulu, HI, April 1-5, 2007). CIDM 2007. IEEE Computer Society, Los Alamitos, CA, 362-367
50. Leo, M., D'Orazio, T., and Spagnolo, P. 2004. Human activity recognition for automatic visual surveillance of wide areas. In *Proceedings of the ACM 2nd international Workshop on Video Surveillance and Sensor Networks* (New York, NY, USA, October 15, 2004). VSSN '04. ACM Press, New York, NY, 124-130.
51. Kang, H. and Cho, S. 2004. Adaptive object tracking using bayesian network and memory. In *Proceedings of the ACM 2nd international Workshop on Video Surveillance and Sensor Networks* (New York, NY, USA, October 15 - 15, 2004). VSSN '04. ACM Press, New York, NY, 104-113.
52. Petrushin, V. A., Wei, G., Ghani, R., and Gershman, A. V. 2005. Multiple sensor integration for indoor surveillance. In *Proceedings of the 6th international Workshop on Multimedia Data Mining: Mining integrated Media and Complex Data* (Chicago, Illinois, August 21 - 21, 2005). MDM '05. ACM Press, New York, NY, 53-60.
53. Kelly, J. 1993. *Artificial intelligence: a modern myth.* E. Horwood, London.
54. Rich, E. and Knight, K. 1991. *Artificial intelligence: a modern myth.* McGraw-Hill, New York.
55. Cozman, F. G. 1998. The Interchange Format for Bayesian Networks (XMLBIF). Retrieved October 9, 2006, From: http://www.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat
56. Kirillov, A. AFORGE. Retrieved July 25, 2007, From Google code: http://code.google.com/p/aforge/
57. Cozman, F. G. 2001 JavaBayes. Retrieved July 25, 2007, From JavaBayes - version 0.346 Bayesian Networks in Java User manual and download: http://www.cs.cmu.edu/~javabayes/Home/

58. Murphy, K. 1998 A Brief Introduction to Graphical Models and Bayesian Networks. Retrieved June 8, 2006, From Graphical Models: http://www.cs.cmu.edu/~javabayes/Home/

59. Fisher, R., Perkins, S., Walker, A. and Wolfart, E. 2003. Connected Components Labelling. Retrieved June 8, 2006, From Image Analysis- Connected Components: http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm

60. Axis 210 Network Camera. Retrieved August 18, 2007, From Axis Communications - AXIS 210 Network Camera: http://www.axis.com/products/cam_210/index.htm

61. Euclidean distance. Retrieved July 1, 2007, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Euclidean_distance

62. Potgieter, A., 2004. *The Engineering of Emergence in Complex Adaptive Systems.* Thesis (PhD). University of Pretoria, Pretoria.

Appendix A

System Design Diagram

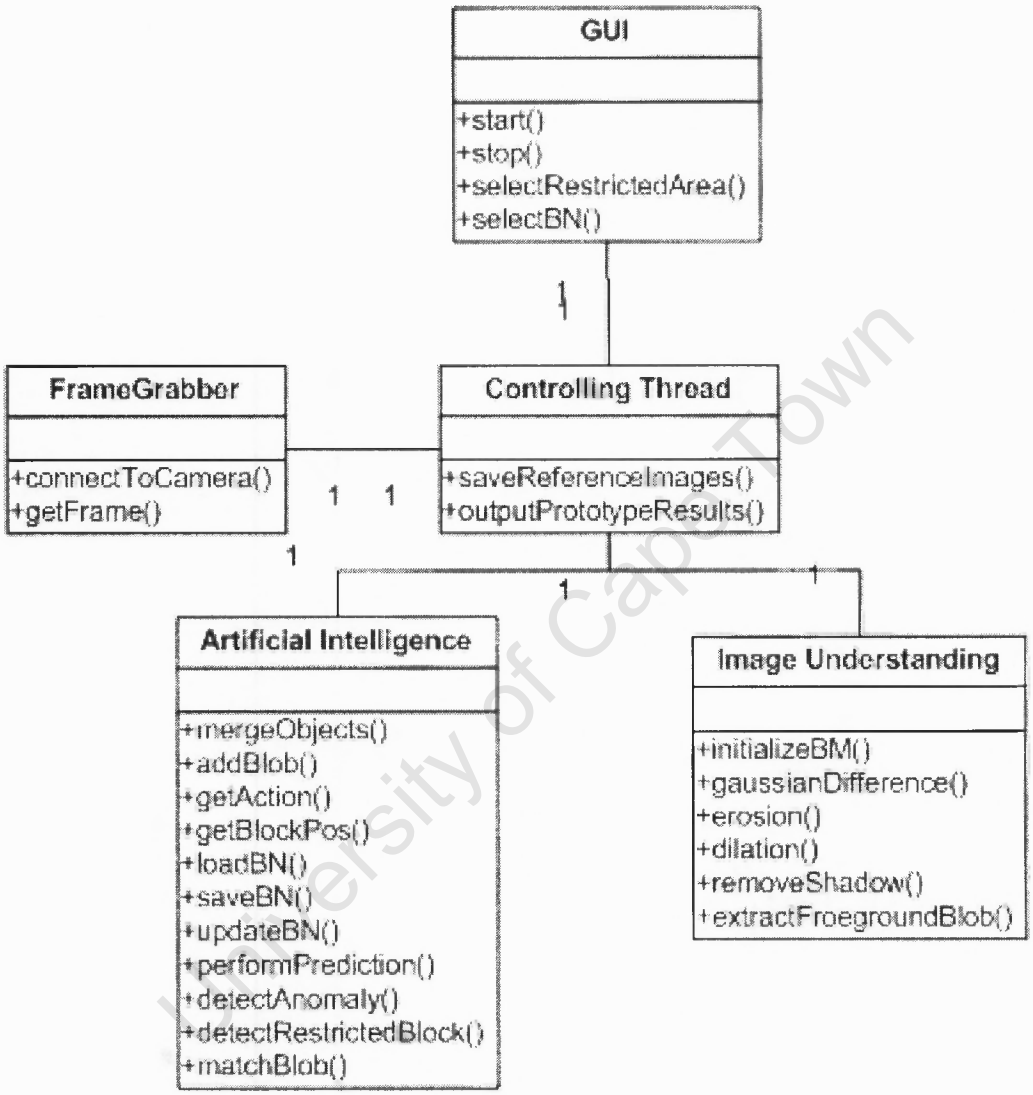


Figure A1: Detailed Class Diagram for the prototype video surveillance system

Appendix B

Detailed Evaluation Test Results

This appendix presents the details and results of each evaluation experiment that was not presented in Chapter 6.

B.1 Euclidean Distance for Merging Small Blobs

The detailed evaluation results for all the 5 tests are shown below.

B.1.1 Test 1

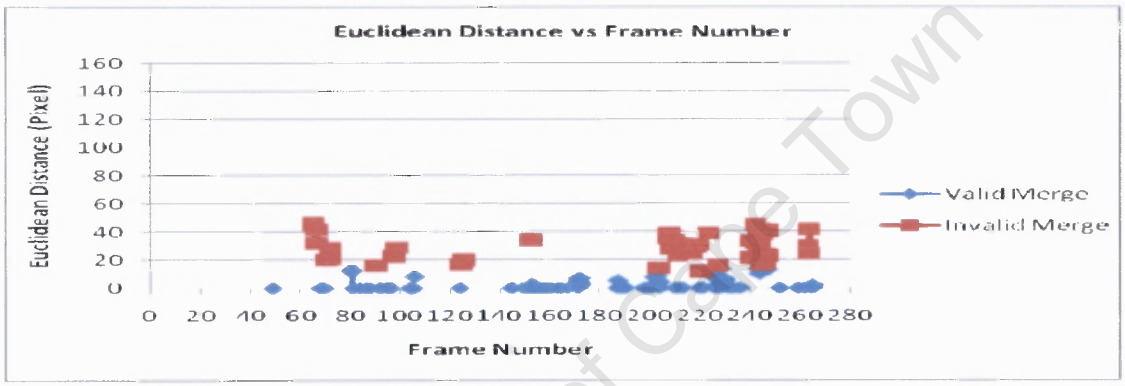


Figure B1.1: The Euclidean distances for valid and invalid small blob merge operations for test 1

Euclidean Distance (Pixel)	Valid (Pixel)	Invalid (Pixel)
Maximum	23.6	46.7
Minimum	0	11.1
Average	2.1	28.9

Table B1.1: The summary of merging small blobs test 1 results

B.1.2 Test 2

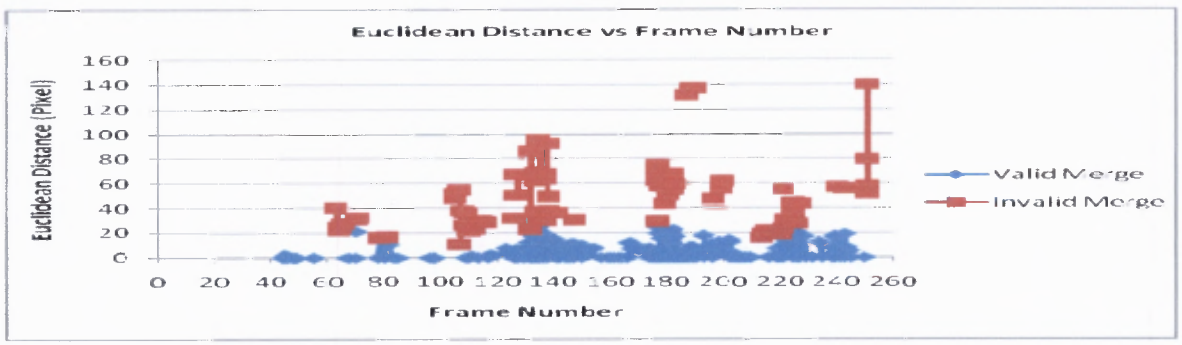


Figure B1.2: The Euclidean distances for valid and invalid small blob merge operations for test 2

Euclidean Distance (Pixel)	Valid (Pixel)	Invalid (Pixel)
Maximum	24.7	140
Minimum	0.0	10.4
Average	4.9	51.8

Table B1.2: The summary of merging small blobs test 2 results

B.1.3 Test 3

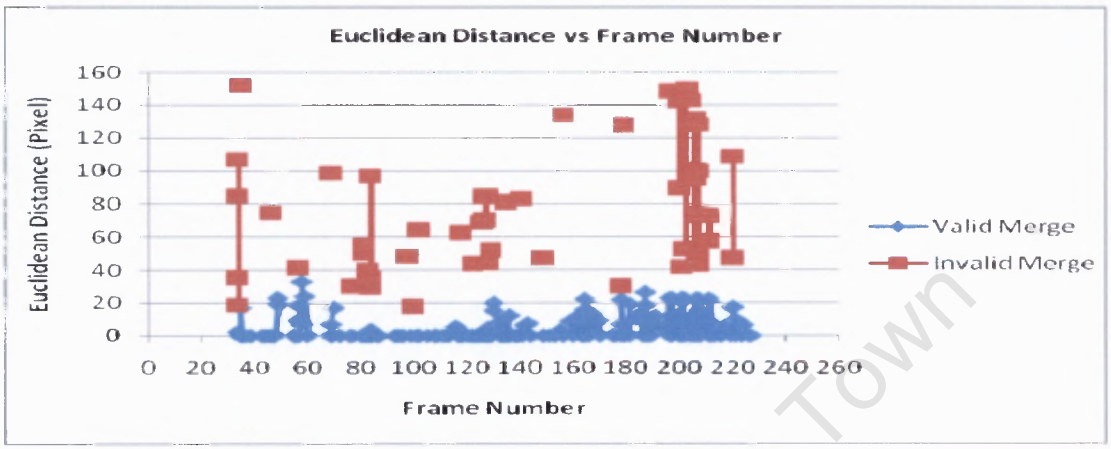


Figure B1.3: The Euclidean distances for valid and invalid small blob merge operations for test 3

Euclidean Distance (Pixel)	Valid (Pixel)	Invalid (Pixel)
Maximum	33.1	151.6
Minimum	0	17.4
Average	4.0	79.1

Table B1.3: The summary of merging small blobs test 3 results

B.1.4 Test 4

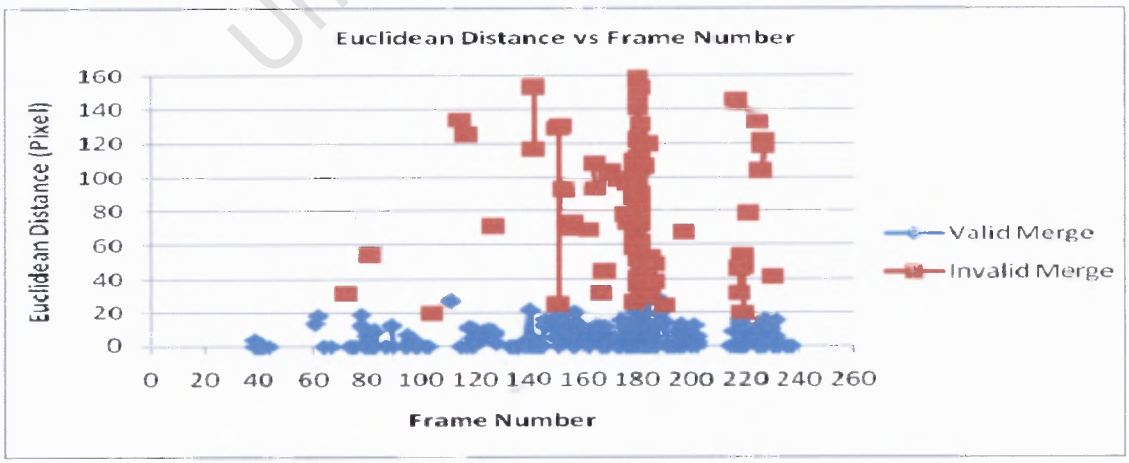


Figure B1.4: The Euclidean distances for valid and invalid small blob merge operations for test 4

Euclidean Distance (Pixel)	Valid	Invalid
Maximum	27.5	159
Minimum	0	19.1
Average	5.7	81.6

Table B1.4: The summary of merging small blobs test 4 results

B.1.6 Test 5

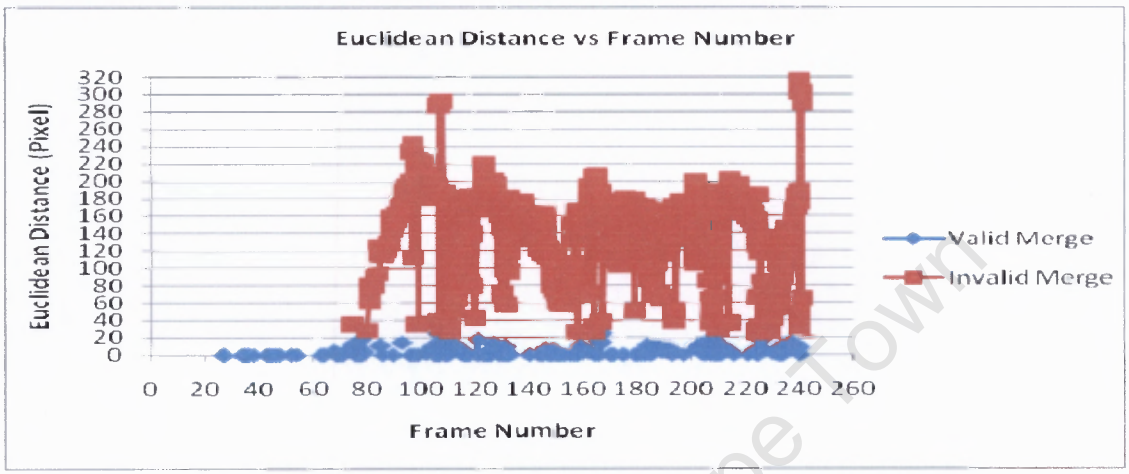


Figure B1.5: The Euclidean distances for valid and invalid small blob merge operations for test 5

Euclidean Distance (Pixels)	Valid	Invalid
Maximum	24.6	314.8
Minimum	0	25
Average	4.7	133.3

Table B1.5: The summary of merging small blobs test 5 results

B.2 Size of Large Noise Regions

The detailed evaluation results for all the 5 tests are shown below.

B.2.1 Test 1

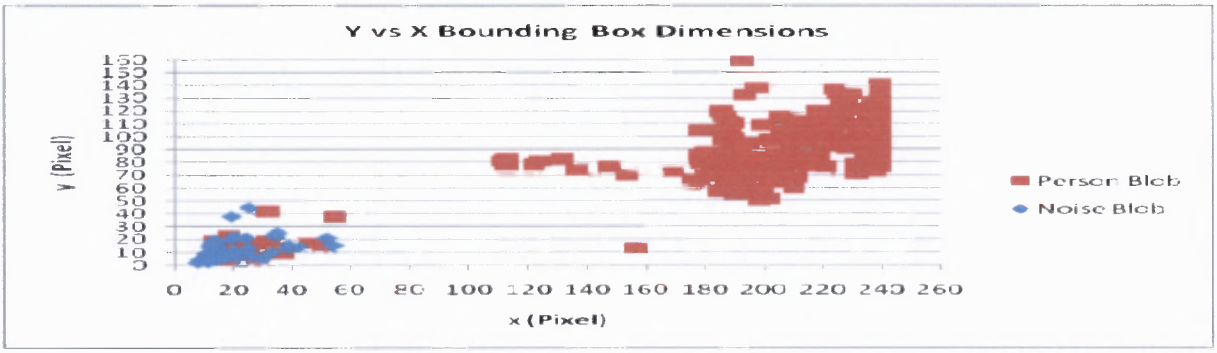


Figure B2.1: The dimension clusters for noise regions and person blobs for test 1

The average dimension of the noise regions' bounding box for Test 1 is 15.1 x 8.0.

B.2.1 Test 2

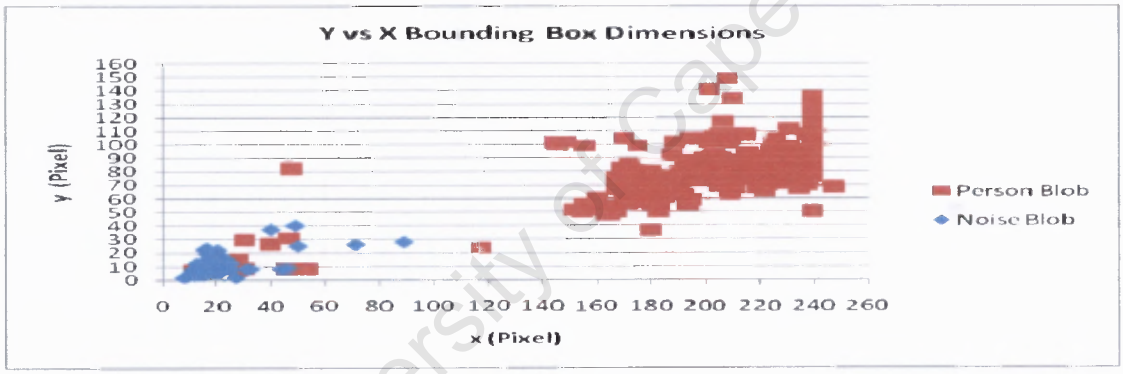


Figure B2.2: The dimension clusters for noise regions and person blobs for test 2

The average dimension of the noise regions' bounding box for Test 2 is 18.1 x 8.9.

B.2.1 Test 3

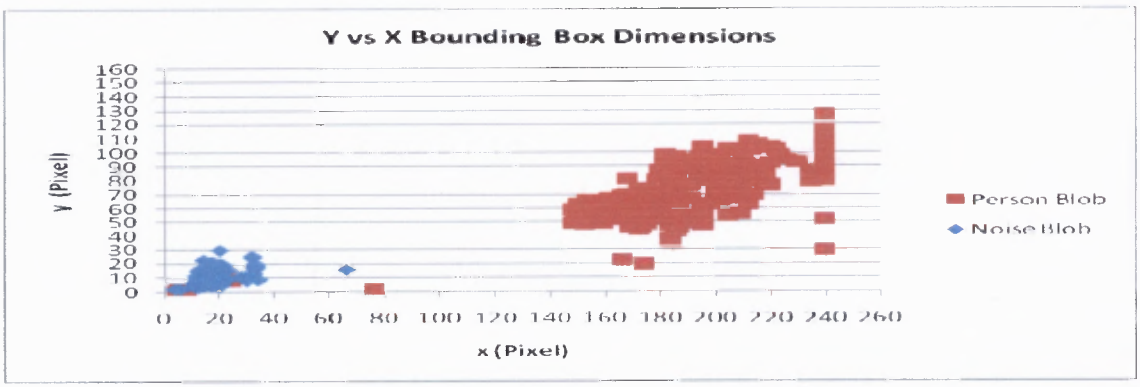


Figure B2.3: The dimension clusters for noise regions and person blobs for test 3

The average dimension of the noise regions' bounding box for Test 3 is 15.3 x 9.0.

B.2.1 Test 4

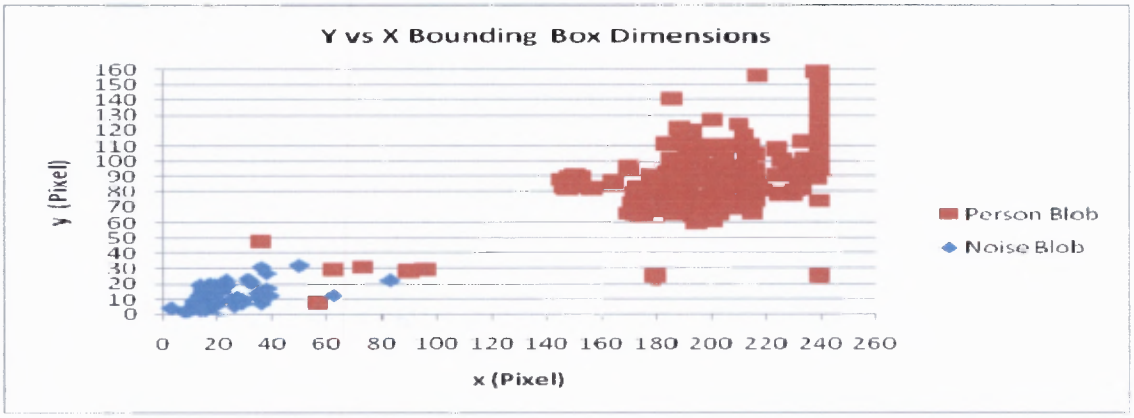


Figure B2.4: The dimension clusters for noise regions and person blobs for test 4

The average dimension of the noise regions' bounding box for Test 4 is 16.6 x 8.5.

B.2.1 Test 5

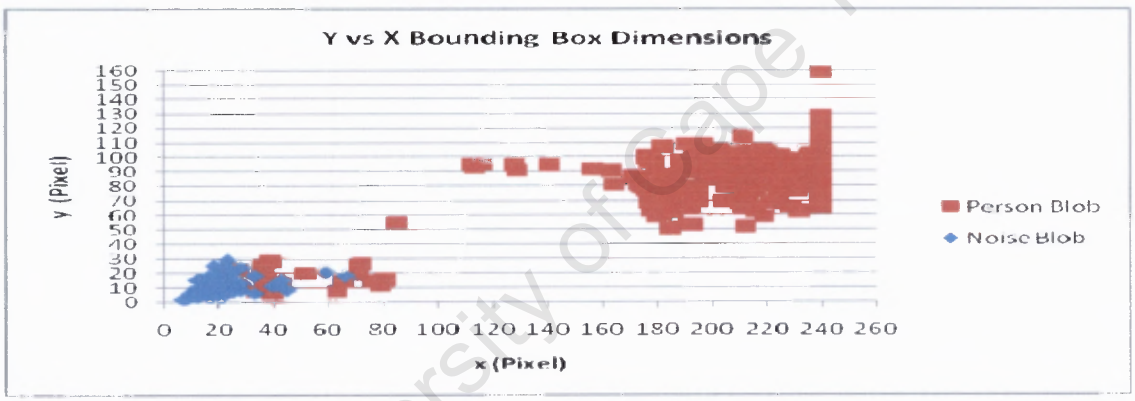


Figure B2.5: The dimension clusters for noise regions and person blobs for test 5

The average dimension of the noise regions' bounding box for Test 5 is 15.9 x 8.0.

B.3 Inference Evaluation Test

B.3.1 Bayesian Network (BN) CPT parameter detail

The BN parameter detail for each node's CPT that was used for the inference evaluation test is shown below:

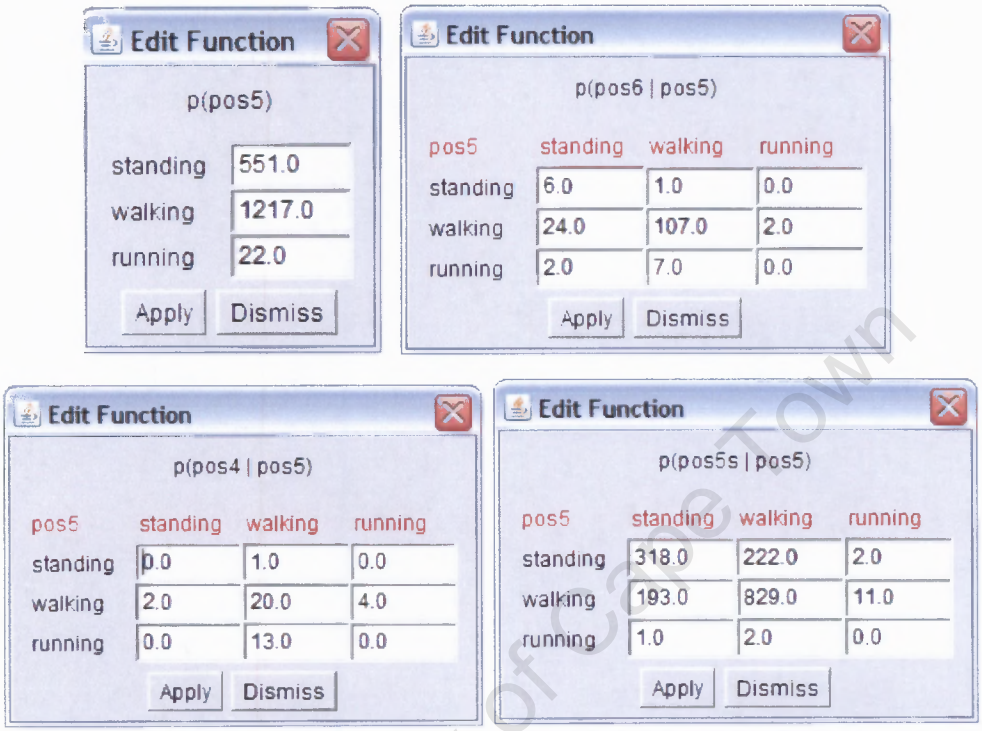


Figure B3.1: CPT parameter details for the BN used in the inference test

B.3.2 Inference Evaluation Test Results

B.3.2.1 Pos4 Tests

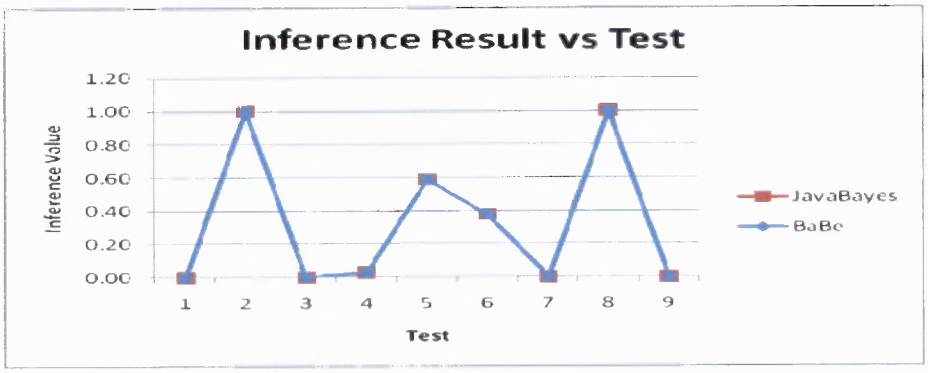


Figure B3.2: Inference Test 1 Result

Query	BaBe Inference	JavaBayes Inference
$P(pos4 = standing \mid pos5 = standing)$	0.00	0.00
$P(pos4 = walking \mid pos5 = standing)$	1.00	1.00
$P(pos4 = running \mid pos5 = standing)$	0.00	0.00
$P(pos4 = standing \mid pos5 = walking)$	0.03	0.03
$P(pos4 = walking \mid pos5 = walking)$	0.59	0.59
$P(pos4 = running \mid pos5 = walking)$	0.38	0.38
$P(pos4 = standing \mid pos5 = running)$	0.00	0.00
$P(pos4 = walking \mid pos5 = running)$	1.00	1.00
$P(pos4 = running \mid pos5 = running)$	0.00	0.00

Table B3.1: Inference test 1 detailed results

B.3.2.2 Pos6 Tests

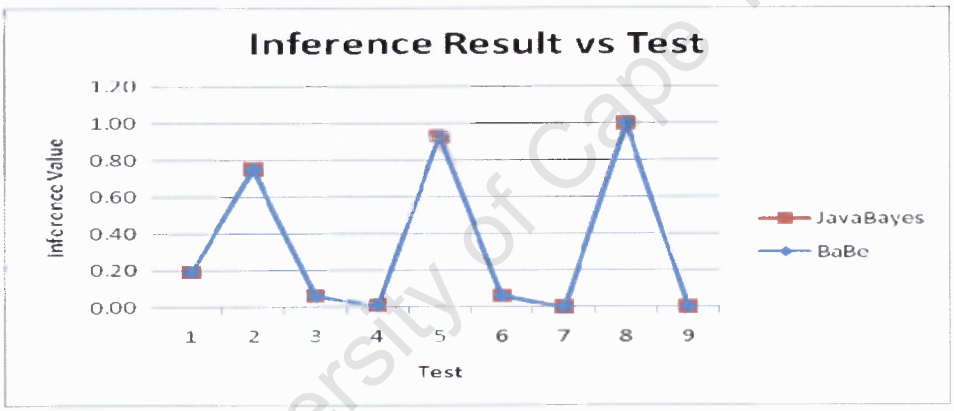


Figure B3.3: Inference Test 2 Results

Query	BaBe Inference	JavaBayes Inference
$P(pos6 = standing \mid pos5 = standing)$	0.19	0.19
$P(pos6 = walking \mid pos5 = standing)$	0.75	0.75
$P(pos6 = running \mid pos5 = standing)$	0.06	0.06
$P(pos6 = standing \mid pos5 = walking)$	0.01	0.01
$P(pos6 = walking \mid pos5 = walking)$	0.93	0.93
$P(pos6 = running \mid pos5 = walking)$	0.06	0.06
$P(pos6 = standing \mid pos5 = running)$	0.00	0.00
$P(pos6 = walking \mid pos5 = running)$	1.00	1.00
$P(pos6 = running \mid pos5 = running)$	0.00	0.00

Table B3.2: Inference test 2 detailed results

B.3.2.3 Pos5s Tests

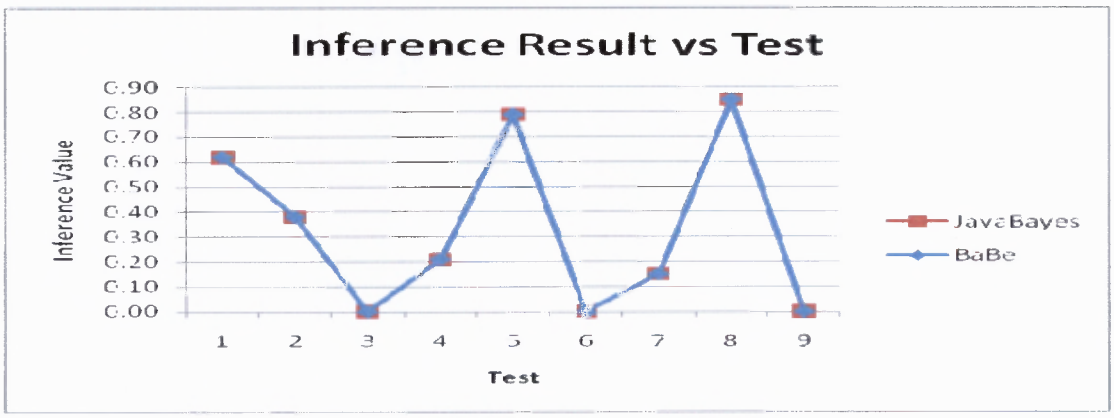


Figure B3.3: Inference Test 3 Result

Query	BaBe Inference	JavaBayes Inference
$P(pos5s = standing \mid pos5 = standing)$	0.62	0.62
$P(pos5s = walking \mid pos5 = standing)$	0.38	0.38
$P(pos5s = running \mid pos5 = standing)$	0.00	0.00
$P(pos5s = standing \mid pos5 = walking)$	0.21	0.21
$P(pos5s = walking \mid pos5 = walking)$	0.79	0.79
$P(pos5s = running \mid pos5 = walking)$	0.00	0.00
$P(pos5s = standing \mid pos5 = running)$	0.15	0.15
$P(pos5s = walking \mid pos5 = running)$	0.85	0.85
$P(pos5s = running \mid pos5 = running)$	0.00	0.00

Table B3.3: Inference test 3 detailed results

B.4 Evaluation of Robustness and Accuracy

B.4.1 Test 1

Evaluation	True Positive	False Positive	True Negative	False Negative
Blob count	106	N/A	N/A	0
Blob tracking	106	N/A	N/A	0
Action recognition	95	N/A	N/A	11
Action prediction	73	N/A	N/A	33
Anomaly Detection	2	12	92	0
Anomaly prediction	0	14	90	2
Restricted area detection	25	0	81	0
Restricted area prediction	26	50	29	1

Table B4.1: Results for robustness and accuracy evaluation test 3

B.4.2 Test 2

Evaluation	True Positive	False Positive	True Negative	False Negative
Blob count	80	N/A	N/A	0
Blob tracking	80	N/A	N/A	0
Action recognition	79	N/A	N/A	1
Action prediction	68	N/A	N/A	12
Anomaly Detection	14	3	23	40
Anomaly prediction	12	1	21	46

Table B4.2: Results for robustness and accuracy evaluation test 2

B.4.3 Test 3

Evaluation	True Positive	False Positive	True Negative	False Negative
Blob count	153	N/A	N/A	0
Blob tracking	153	N/A	N/A	0
Action recognition	141	N/A	N/A	12
Action prediction	108	N/A	N/A	45
Anomaly Detection	41	7	91	14
Anomaly prediction	33	25	76	19
Restricted area detection	28	0	125	0
Restricted area prediction	31	44	78	0

Table B4.3: Results for robustness and accuracy evaluation test 3

B.4.4 Test 4

Evaluation	True Positive	False Positive	True Negative	False Negative
Blob count	236	N/A	N/A	0
Blob tracking	236	N/A	N/A	0
Action recognition	235	N/A	N/A	1
Action prediction	190	N/A	N/A	46
Anomaly Detection	30	1	205	0
Anomaly prediction	18	13	191	14
Restricted area detection	55	0	181	0
Restricted area prediction	58	93	85	0

Table B4.4: Results for robustness and accuracy evaluation test 4

B.4.5 Test 5

Evaluation	True Positive	False Positive	True Negative	False Negative
Blob count	179	N/A	N/A	0
Blob tracking	179	N/A	N/A	0
Action recognition	178	N/A	N/A	1
Action prediction	172	N/A	N/A	7
Anomaly Detection	8	1	169	1
Anomaly prediction	9	17	152	1
Restricted area detection	0	0	179	0
Restricted area prediction	9	9	179	0

Table B4.5: Results for robustness and accuracy evaluation test 5

B.5 Evaluation of Robustness and Accuracy on PETS2002

B.5.1 Test Results

Evaluation	True Positive	False Positive	True Negative	False Negative
Blob count	847	4	53	497
Blob tracking	843	N/A	N/A	5
Action recognition	813	N/A	N/A	35
Action prediction	811	N/A	N/A	37
Anomaly Detection	84	30	728	6
Anomaly prediction	78	60	703	7

Table B5.1: Results for robustness and accuracy evaluation test on the PETS 2002 dataset 1

Appendix C

Image Understanding Concepts

C.1 Image Understanding Basics

C.1.1 Pixel

A pixel, or picture element, is the smallest part of an image or frame. In other words, according to Gonzalez and Woods [47] pixels are the elements that make up an image. A pixel can have several colour formats, including:

- Grey-scale – shades of grey;
- RGB – 3 colour values stored per pixel, red, green, blue;
- YUV – Y is the luminance (brightness). U and V are colour values;
- HVS – H is hue, S is saturation, V is the value or brightness.

Good sources for further reading about pixels and the different colour formats are [25] and [47].

C.1.2 Image (Frame)

An image, or frame, is the visual representation of an environment. An image is usually produced by a digital camera, but may be produced by some other form of digitization e.g. a scanner. An image is made up of several pixels. Traditionally, an image is represented as an $N \, x \, M$ matrix comprising of pixels, as shown below:

$$image = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}, \text{ where the dots are pixels}$$

In addition, the dimensions of the pixels that make up an image are used as a representation of the image resolution (width x height). In this dissertation, the term frame is used when referring to an image in a sequence of images.

C.1.3 Mean

The mean value (μ) is the average value of a population N consisting of variables $x1, x2 \ldots xN$ is defined by Keller and Warrack [28] as:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{N} \tag{C.1}$$

When the population size is infinitely large the Expected Value, E(X), or weighed average has to be used instead of the mean. For a random variable X, with values x_i and probability $p(x_i)$ of x_i occurring, Keller and Warrack [28] define the Expected Value as:

$$E(X) = \sum_{all\ x_i} x_i * p(x_i) \tag{C.2}$$

C.1.4 Variance

The variance (σ^2) is a measure of how dispersed the values of a population are from the population mean (μ), and is defined by Keller and Warrack [28] as

$$\sigma^2 = E[(X - \mu)^2] = \sum_{all\ x_i} (x_i - \mu)^2\, p(x_i) \tag{C.3}$$

The standard deviation is a way of getting the variability of a population around its mean in the same units and defined by Keller and Warrack [28] as the positive square root of the variance of a population.

C.1.5 Gaussian Distribution

First introduced by Abraham de Moivre in an article in 1734, the Gaussian distribution or Normal distribution is a "continuous probability distribution" [26], [28]. The distribution is defined by two parameters, namely the mean and variance, and is best represented graphically by a Gaussian Function or Probability Density Function (PDF) which is shown in Figure 1 on the next page:

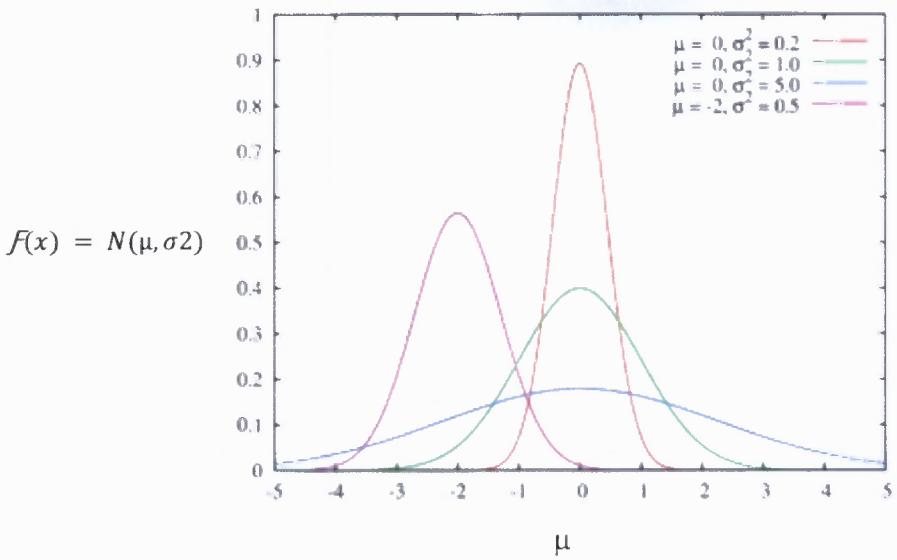


Figure C.1: Gaussian Probability Density Functions (PDF) with varying mean and variance values [26]

The PDF equation for the Gaussian distribution is shown below:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{C.4}$$

where σ is the standard deviation, μ is the mean and x is the current value. [26] and [28].

According to [28], a Gaussian distribution is fully determined once the μ and σ^2 variables are specified. Another important feature of the Gaussian distribution is the fact that, about 68% of values drawn from a standard normal distribution are within 1 standard deviation away from the mean; about 95% of the values are within two standard deviations and about 99.7% lie within 3 standard deviations. This is known as the "68-95-99.7 rule" or the "empirical rule." [26]

C.1.6 Metric

A metric is a measurement which is generally used as a standard of measure for comparison or evaluation purposes.

C.1.7 Mahalanobis Distance

Introduced by P. C. Mahalanobis in 1936, the Mahalanobis distance is a distance metric which is based on the correlation between variable sets [27]. The formal definition, adapted from [27], of the Mahalanobis distance is shown on the next page:

$$D_M(x) = \sqrt{(x-\mu)^T \sum^{-1} (x-\mu)} \quad \text{(C.5)}$$

where, $(\ )^T$ is the matrix transpose and $(\ )^{-1}$ is the matrix inverse and D_M is the Mahalanobis distance from a group of values with mean $\mu = (\mu_1, \mu_1, \mu_1, \ldots, \mu_p)^T$ and the covariance matrix Σ for a multivariate vector $x = (x, x, x, \ldots, x_p)^T$.

This metric can be used to determine the similarity or difference of an unknown set to a known one. In this dissertation the Mahalanobis distance is used to measure the distance between the pixels in the current frame and the pixels in the background model.

C.1.8 Euclidean Distance

According to [61] the Euclidean distance, or Euclidean metric, is a measure of the shortest distance between two or more points. It represents the distance measurement that could be made if a ruler was used to measure the distance between the points. The Euclidean distance measure can be applied to points with multiple variables. The formula for working out the Euclidean distance, adopted from [61], between two points with n variables, namely $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$ is defined below:

$$\sqrt{(p_1 - p_2)^2 + (p_1 - p_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad \text{(C.6)}$$

During this dissertation the Euclidean distance metric was extensively used in the prototype surveillance system to calculate the distance between groups of pixels.

C1.9 Image Morphology

According to Gonzalez and Woods [47], image morphology is the process of altering an image in either form or structure and is based on set theory. Basic image morphology is achieved by applying either the *dilation* or *erosion* fundamental operations. These operations can be combined to get more complex morphological operations, including *opening* and *closing*, which can then be used to enhance an image or remove random noise from an image. The following discussion on morphological operators is adapted from [47].

In the discussion of the morphological operators the following is assumed:

- Z^2 is a 2-D integer space
- A and B are sets in Z^2 whose elements have (x, y) co-ordinates which are relative to an image. i.e. $b \in B$, such that $b = (b_1, b_2)$ where b_1 and b_2 are the (x, y) co-ordinates
- Set A is commonly the set of all pixels in the image that is being operated on
- Set B is commonly referred to as the structuring element with the most common shapes being squares, circles and crosses
- The translation of set B by $x = (x_1, x_2)$ is denoted as $(B)_x$
- The reflection of set B is denoted as $\hat{B}$
- $\emptyset$ denotes an empty set

1. Dilation

A dilation operation is used to increase the size of the regions present in an image or to bridge small gaps within the regions. It is denoted as $A \oplus B$ and is defined by Gonzalez and Woods [47] as:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \tag{C.7}$$

The resultant shape of a region after applying the dilation operator is dependent on the structuring element that is used.

2. Erosion

An erosion operation is used to reduce the size of regions that are present in a region and for eliminating regions that are irrelevant. It is denoted as $A \ominus B$ and is defined by Gonzalez and Woods [47] as:

$$A \ominus B = \{z \mid (B) \subseteq A\} \tag{C.8}$$

The resultant shape of a region after applying the erosion operator is dependent on the structuring element that is used.

3. Opening

An opening operation is a combination of the two fundamental morphological operators and is generally used for smoothing the contour of an object by breaking regions that are connected by narrow spaces and eliminating thin protrusions [47]. The opening operator firstly applies an erosion operation followed by a dilation operation. It is denoted as $A \bullet B$ and is defined by Gonzalez and Woods [47] as:

$$A \bullet B = \big((A \ominus B) \oplus B\big) \tag{C.9}$$

4. Closing

A closing operation is also a combination of the two fundamental morphological operators and likewise it is generally used for smoothing the contour of an object. However contrary to an opening operation, smoothing is achieved by fusing regions that are separated by narrow gaps, eliminating small holes within the regions and filling gaps in the contour [47]. The closing operator firstly applies a dilation operation followed by an erosion operation. It is denoted as $A \circ B$ and is defined by Gonzalez and Woods [47] as:

$$A \circ B = \big((A \oplus B) \ominus B\big) \tag{C.10}$$

Appendix D

Surveillance System Evaluation Techniques

D.1 Evaluation of Surveillance Systems

Evaluation of surveillance systems has attracted a lot of interest from researchers in an effort to make accurate systems that can be used in real world environments as opposed to controlled test environments. Evaluation metrics are commonly defined and used to evaluate surveillance systems. Besides evaluating the correctness and accuracy of video surveillance systems, such evaluation metrics can also provide a way for several video surveillance systems to be compared.

A popular method of comparing various surveillance systems is to use common testing data such as the one provided by Performance and Evaluation of Tracking Systems (PETS) [41]. In addition, PETS [41] provides ground truth data for most of their test data so that comparisons of surveillance system evaluations can be made. In addition, an online facility is also available, where surveillance data can be submitted for comparison by experts in the field using standard evaluation metrics. However such a comparison method assumes that the surveillance systems that are being compared have been developed with common objectives or goals.

D.1.1 Ground Truth Data

Ground truth data is data that accurately represents events and objects in an environment and can be used as a benchmark for comparison by video surveillance systems. Ground truth data is typically extracted manually by a person, usually an expert in the environment that is being monitored. More precisely, the person or expert monitors the entire image sequence so that an accurate representation of the environment is obtained. Ground truth data can contain simple data such as the actual number of blobs in the environment or complex data such as the actual location of the blobs, the centre of the each blob's bounding box or the actions that the blobs perform. The detail and amount of ground truth data that is extracted is dependent on the kind of evaluation that is being performed.

The extraction of ground truth data is a very laborious and time consuming process thus may be error prone. This fact is highlighted in [40] and [42]. In order to avoid errors more than one person should be involved in the extraction of ground truth data. Recently, some effort has been made in automatically generating ground truth data such as the work of [40]. Ellis [40] suggests the use of semi-automated generation of ground truth data with manual intervention in the case of errors. However, such a technique would be biased towards algorithms that are similar to the one used in the semi-automated generation. Another alternative could be the use of synthetic video, but according to Ellis [40] this type of data can be prone to error when the location of noise in the synthetic video is known.

In order to compare video surveillance data with the ground truth data for a particular environment, the following quantity measurements need to be recorded:

1. ***True Positives (TP)*** The data that is present in both the ground truth data and the video surveillance data
2. ***False Positives (FP)*** The data that is not present in the ground truth data but is present in the video surveillance data
3. ***True Negatives (TN)*** The data that is correctly missing in both the ground truth data and the surveillance data
4. ***False Negatives (FN)*** The data that is missing in the video surveillance data but is present in the ground truth data.

D.1.2 Evaluation Metrics

Several evaluation metrics have appeared recently in literature as focus shifts on evaluation of surveillance systems. According to Ellis [40] a widely used metric for evaluation of surveillance systems is the processing time of the system, but as a result of advances in hardware development such a metric will soon not be sufficient enough to fully evaluate a surveillance system. Some evaluation metrics found in literature are presented below:
Ellis [40] defines the following evaluation metrics:

1. ***detection rate (sensitivity)*** $TP/(TP + FN)$ (D.1)
2. ***specificity*** $TN/(TN + FP)$ (D.2)
3. ***accuracy*** $(TN + TP)/N$, where N is total number of measurements (D.3)

4. ***positive predictive value*** $TP/(TP + FP)$ (D.4)
5. ***false negative rate*** $FN/(TP + FN)$ (D.5)
6. ***false positive rate*** $FP/(FP + TN)$ (D.6)
7. ***negative predictive value*** $TN/(TN + FN)$ (D.7)

Bashir and Porikli [22] define the following additional evaluation metrics:

8. ***tracker detection rate*** TP/TG , where TG is the total number of measurements in the ground truth data (D.8)
9. ***false alarm rate*** $FP/(TP + FP)$ (D.9)

D.1.3 Evaluation Areas

There are several areas in a surveillance system that can be evaluated using the above metrics, these are discussed below.

1. Segmentation of blobs

The correct segmentation of blobs is a critical task in a video surveillance system because the tracking and activity recognition systems are dependent on it. Hence the evaluation of correctly segmented blobs is very important. The distinction between random noise regions and actual foreground pixels forms the basis of the evaluation task. In addition, the formation of bounding boxes and their centroid can be evaluated for correctness when optical flow or bounding boxes are used for tracking. Evaluation of the correctness in segmentation of blobs will show the robustness and accuracy of the surveillance system in extracting the real objects present in the environment.

2. Blob Counts

The number of blobs detected in an environment can be counted and used for comparison with other systems working on the same data. This is good way of determining whether environmental factors (light illumination, moving background objects etc), occlusion, merging of objects, ghosts and shadows are dealt with properly. Furthermore, in an environment where the density of people is monitored, e.g. sports matches, this can be a critical task and deserves a thorough evaluation.

3. Blob Classification

The identification of various objects in an environment where several types of objects are present is rapidly becoming an important surveillance task. Initially, environments with few objects were considered, such as people and cars [4] and [7], but now environments with several different objects are being monitored such as abandoned baggage at public places [43], [44], [45] and [46]. These various objects need to be distinguished as focus on security is magnified by the global community. Thus evaluating the correct classification of objects is an important task.

4. Tracking of Blobs

In order for analysis of blob behaviour to be done the tracking of blobs must be very accurate. If this is not the case the analysis may lead to false assumptions. Hence, evaluation of the tracking part of a surveillance system is critical in order to ensure correctness, robustness and accuracy of the tracking system.

5. Action Recognition

Action recognition has become the norm in most surveillance systems and its interpretation may lead to serious action or allegations brought onto individuals that are being monitored. That fact alone stresses the need for accurate action recognition because nothing is worse than falsely accusing a person. Hence, the action recognition part has a large scope for evaluation.

Appendix E

XMLBIF Sample

E.1 Sample Iteration of an XMLBIF file used during the research

```
<?xml version="1.0"?>
<!--Name:Agents Lab-->
<!--DTD for the BIF format-->
<!DOCTYPE BIF [
        <!ELEMENT BIF ( NETWORK )*>
        <!ELEMENT PROPERTY (#PCDATA)>
        <!ELEMENT TYPE (#PCDATA)>
        <!ELEMENT VALUE (#PCDATA)>
        <!ELEMENT NAME (#PCDATA)>
        <!ELEMENT NETWORK
          ( NAME, ( PROPERTY | VARIABLE |
PROBABILITY )* )>
        <!ELEMENT VARIABLE ( NAME, TYPE, (
VALUE |  PROPERTY )* ) >
        <!ELEMENT PROBABILITY
          ( FOR | GIVEN | TABLE | ENTRY | DEFAULT |
PROPERTY )* >
        <!ELEMENT FOR (#PCDATA)>
        <!ELEMENT GIVEN (#PCDATA)>
        <!ELEMENT TABLE (#PCDATA)>
        <!ELEMENT DEFAULT (TABLE)>
        <!ELEMENT ENTRY ( VALUE* , TABLE )>
]>
<BIF VERSION="0.3">
 <NETWORK>
  <NAME>block0Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos0</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos1</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos4</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos5</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos0s</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <!--Probability distributions-->
  <DEFINITION>
   <FOR>pos0</FOR>
   <TABLE>0 5 1</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos1</FOR>
   <GIVEN>pos0</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos4</FOR>
   <GIVEN>pos0</GIVEN>
   <TABLE>0 0 0 0 1 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos5</FOR>
   <GIVEN>pos0</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos0s</FOR>
   <GIVEN>pos0</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
 </NETWORK>
 <NETWORK>
  <NAME>block1Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos1</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos0</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos2</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos4</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos5</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
```

```
  <VARIABLE>
   <NAME>pos6</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos1s</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <!--Probability distributions-->
  <DEFINITION>
   <FOR>pos1</FOR>
   <TABLE>0 6 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos0</FOR>
   <GIVEN>pos1</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos2</FOR>
   <GIVEN>pos1</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos4</FOR>
   <GIVEN>pos1</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos5</FOR>
   <GIVEN>pos1</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos6</FOR>
   <GIVEN>pos1</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos1s</FOR>
   <GIVEN>pos1</GIVEN>
   <TABLE>0 0 0 0 2 0 0 0 0</TABLE>
  </DEFINITION>
 </NETWORK>
 <NETWORK>
  <NAME>block2Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos2</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos1</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos3</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos5</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos6</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos7</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos2s</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <!--Probability distributions-->
  <DEFINITION>
   <FOR>pos2</FOR>
   <TABLE>148 27 2</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos1</FOR>
   <GIVEN>pos2</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos3</FOR>
   <GIVEN>pos2</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos5</FOR>
   <GIVEN>pos2</GIVEN>
   <TABLE>1 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos6</FOR>
   <GIVEN>pos2</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos7</FOR>
   <GIVEN>pos2</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos2s</FOR>
   <GIVEN>pos2</GIVEN>
   <TABLE>127 11 0 21 4 0 0 1 0</TABLE>
  </DEFINITION>
 </NETWORK>
 <NETWORK>
  <NAME>block3Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos3</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos2</NAME>
```

```
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos6</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos7</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos3s</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <!--Probability distributions-->
 <DEFINITION>
  <FOR>pos3</FOR>
  <TABLE>2 2 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos2</FOR>
  <GIVEN>pos3</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos6</FOR>
  <GIVEN>pos3</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos7</FOR>
  <GIVEN>pos3</GIVEN>
  <TABLE>0 1 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos3s</FOR>
  <GIVEN>pos3</GIVEN>
  <TABLE>1 0 0 1 1 0 0 0 0</TABLE>
 </DEFINITION>
</NETWORK>
<NETWORK>
 <NAME>block4Def</NAME>
 <!--Variables-->
 <VARIABLE>
  <NAME>pos4</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (0, 10)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos0</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos1</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos5</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos8</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos9</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos4s</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <!--Probability distributions-->
 <DEFINITION>
  <FOR>pos4</FOR>
  <TABLE>122 288 15</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos0</FOR>
  <GIVEN>pos4</GIVEN>
  <TABLE>0 0 0 0 0 1 0 1 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos1</FOR>
  <GIVEN>pos4</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos5</FOR>
  <GIVEN>pos4</GIVEN>
  <TABLE>0 6 0 0 32 10 0 1 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos8</FOR>
  <GIVEN>pos4</GIVEN>
  <TABLE>0 0 0 0 2 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos9</FOR>
  <GIVEN>pos4</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos4s</FOR>
  <GIVEN>pos4</GIVEN>
  <TABLE>90 25 0 31 172 2 0 10 0</TABLE>
 </DEFINITION>
</NETWORK>
<NETWORK>
 <NAME>block5Def</NAME>
 <!--Variables-->
 <VARIABLE>
  <NAME>pos5</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (0, 10)</PROPERTY>
 </VARIABLE>
```

Appendix E: XMLBIF Sample

```
<VARIABLE>
 <NAME>pos0</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos1</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos2</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos4</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos6</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos8</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos9</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos10</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<VARIABLE>
 <NAME>pos5s</NAME>
 <OUTCOME>standing</OUTCOME>
 <OUTCOME>walking</OUTCOME>
 <OUTCOME>running</OUTCOME>
 <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<!--Probability distributions-->
<DEFINITION>
 <FOR>pos5</FOR>
 <TABLE>551 1217 22</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos0</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos1</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 0 0 0 3 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos2</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos4</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 2 0 1 20 13 0 4 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos6</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>6 24 2 1 107 7 0 2 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos8</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos9</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 2 0 0 0 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos10</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
 <FOR>pos5s</FOR>
 <GIVEN>pos5</GIVEN>
 <TABLE>318 193 1 222 829 2 2 11 0</TABLE>
</DEFINITION>
</NETWORK>
<NETWORK>
 <NAME>block6Def</NAME>
 <!--Variables-->
 <VARIABLE>
  <NAME>pos6</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (0, 10)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos1</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos2</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos3</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos5</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
```

```
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos7</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos9</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos10</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos11</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos6s</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <!--Probability distributions-->
 <DEFINITION>
  <FOR>pos6</FOR>
  <TABLE>876 1590 13</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos1</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos2</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 0 0 0 0 1 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos3</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos5</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>5 25 1 4 103 4 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos7</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 10 0 0 16 2 0 1 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos9</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos10</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 3 0 0 3 1 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos11</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>0 0 0 0 1 1 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos6s</FOR>
  <GIVEN>pos6</GIVEN>
  <TABLE>500 329 0 366 1084 0 2 9 0</TABLE>
 </DEFINITION>
</NETWORK>
<NETWORK>
 <NAME>block7Def</NAME>
 <!--Variables-->
 <VARIABLE>
  <NAME>pos7</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (0, 10)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos2</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos3</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos6</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos10</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos11</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos7s</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <!--Probability distributions-->
 <DEFINITION>
  <FOR>pos7</FOR>
  <TABLE>42 127 2</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos2</FOR>
  <GIVEN>pos7</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
```

```
  <DEFINITION>
   <FOR>pos3</FOR>
   <GIVEN>pos7</GIVEN>
   <TABLE>0 0 0 0 1 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos6</FOR>
   <GIVEN>pos7</GIVEN>
   <TABLE>0 9 0 0 12 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos10</FOR>
   <GIVEN>pos7</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos11</FOR>
   <GIVEN>pos7</GIVEN>
   <TABLE>0 1 0 2 5 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos7s</FOR>
   <GIVEN>pos7</GIVEN>
   <TABLE>12 21 0 30 74 0 0 1 0</TABLE>
  </DEFINITION>
 </NETWORK>
 <NETWORK>
  <NAME>block8Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos8</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos4</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos5</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos9</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos8s</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <!--Probability distributions-->
  <DEFINITION>
   <FOR>pos8</FOR>
   <TABLE>5 23 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos4</FOR>
   <GIVEN>pos8</GIVEN>
   <TABLE>0 0 0 0 5 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos5</FOR>
   <GIVEN>pos8</GIVEN>
   <TABLE>0 0 0 0 1 2 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos9</FOR>
   <GIVEN>pos8</GIVEN>
   <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos8s</FOR>
   <GIVEN>pos8</GIVEN>
   <TABLE>2 2 0 3 8 0 0 0 0</TABLE>
  </DEFINITION>
 </NETWORK>
 <NETWORK>
  <NAME>block9Def</NAME>
  <!--Variables-->
  <VARIABLE>
   <NAME>pos9</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (0, 10)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos4</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos5</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos6</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos8</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos10</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <VARIABLE>
   <NAME>pos9s</NAME>
   <OUTCOME>standing</OUTCOME>
   <OUTCOME>walking</OUTCOME>
   <OUTCOME>running</OUTCOME>
   <PROPERTY>position = (5, 0)</PROPERTY>
  </VARIABLE>
  <!--Probability distributions-->
  <DEFINITION>
   <FOR>pos9</FOR>
   <TABLE>5 13 0</TABLE>
  </DEFINITION>
  <DEFINITION>
   <FOR>pos4</FOR>
   <GIVEN>pos9</GIVEN>
```

```
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos5</FOR>
  <GIVEN>pos9</GIVEN>
  <TABLE>0 1 0 0 1 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos6</FOR>
  <GIVEN>pos9</GIVEN>
  <TABLE>0 0 0 0 1 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos8</FOR>
  <GIVEN>pos9</GIVEN>
  <TABLE>0 0 0 0 1 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos10</FOR>
  <GIVEN>pos9</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos9s</FOR>
  <GIVEN>pos9</GIVEN>
  <TABLE>4 0 0 1 9 0 0 0 0</TABLE>
 </DEFINITION>
</NETWORK>
<NETWORK>
 <NAME>block10Def</NAME>
 <!--Variables-->
 <VARIABLE>
  <NAME>pos10</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (0, 10)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos5</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos6</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos7</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos9</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos11</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos10s</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <!--Probability distributions-->
 <DEFINITION>
  <FOR>pos10</FOR>
  <TABLE>33 43 1</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos5</FOR>
  <GIVEN>pos10</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos6</FOR>
  <GIVEN>pos10</GIVEN>
  <TABLE>0 3 0 0 3 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos7</FOR>
  <GIVEN>pos10</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos9</FOR>
  <GIVEN>pos10</GIVEN>
  <TABLE>0 0 0 0 2 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos11</FOR>
  <GIVEN>pos10</GIVEN>
  <TABLE>0 0 0 0 0 0 0 0 0</TABLE>
 </DEFINITION>
 <DEFINITION>
  <FOR>pos10s</FOR>
  <GIVEN>pos10</GIVEN>
  <TABLE>21 9 0 12 23 0 0 0 0</TABLE>
 </DEFINITION>
</NETWORK>
<NETWORK>
 <NAME>block11Def</NAME>
 <!--Variables-->
 <VARIABLE>
  <NAME>pos11</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (0, 10)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos6</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos7</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos10</NAME>
  <OUTCOME>standing</OUTCOME>
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
 </VARIABLE>
 <VARIABLE>
  <NAME>pos11s</NAME>
  <OUTCOME>standing</OUTCOME>
```

```
  <OUTCOME>walking</OUTCOME>
  <OUTCOME>running</OUTCOME>
  <PROPERTY>position = (5, 0)</PROPERTY>
</VARIABLE>
<!--Probability distributions-->
<DEFINITION>
  <FOR>pos11</FOR>
  <TABLE>81 109 1</TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>pos6</FOR>
  <GIVEN>pos11</GIVEN>
  <TABLE>0 2 0 0 1 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>pos7</FOR>
  <GIVEN>pos11</GIVEN>
  <TABLE>0 0 0 0 3 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>pos10</FOR>
  <GIVEN>pos11</GIVEN>
  <TABLE>0 1 0 0 2 0 0 0 0</TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>pos11s</FOR>
  <GIVEN>pos11</GIVEN>
  <TABLE>53 24 0 26 69 0 0 1 0</TABLE>
</DEFINITION>
</NETWORK>
</BIF>
```