University of Edinburgh
Division of Informatics

Intelligent Intruder Detection from a Video Sequence

4th Year Project Report
Computer Science

Gavin Dutch

May 29, 2003

Abstract

This paper shall discuss an intelligent approach to intruder detection and the necessary low-level processing features required to support it. It presents a system that is capable of identifying intruders from video sequences producing data at a rate of 3.5MBytes/sec. The system is particularly tolerant of illumination changes and other background movement resulting in a low false detection rate. Experimental results show that the identification of intruders is highly accurate.

Acknowledgements

I would like to thank my supervisor Robert Fisher for his guidance and support throughout the duration of the project. I would also like to thank Gregor Miller and Robert Patrick for their review and comments on this document.

Contents

1. Introduction

The recognition of significant events in a video surveillance system has been a topic of much debate and discussion within academic and commercial circles for many years now. The challenge of producing an accurate yet efficient system that is capable of identifying significant events and recording them, from video sequences producing 18 Mb/s is rooted in many facets of Artificial Intelligence, Computer Science, Computer Vision and Mathematics. Several techniques have been pioneered. However, researchers are still in search of a system, which within the bounds of current computing power, will accurately identify only significant targets in a real-time situation.

Video surveillance is the term used to encapsulate the process of analysing a stream of video data, recognising a significant event then acting appropriately on that event. Most contemporary systems use a closed-circuit TV network and a human operator to recognise and record events. The problem with this is that human operators are fallible and often miss events whilst distracted or bored.

The aim of this project is to build upon some of the currently available techniques to produce a system capable of identifying, with a reasonable degree of accuracy, only significant events. Insignificant events such as those produced by illumination changes, camera noise or environmental fluctuations (e.g. trees swaying in the wind) are specifically to be ignored.

1.1 Project Specification

The project requires processing to be done in a sequence of stages.

- Primarily the system must perform pixel wise analysis on each frame to find any significantly changed pixels ignoring those that have changed because of camera noise.

- Secondly the system must group these interesting pixels and perform analysis to determine if they are due to an interesting event

Full video streams generate data at around 18 Mbytes/sec however to reduce the processing requirements of the system a lower data rate could be tolerated. This lower data rate may be accomplished by reducing the frame rate or resolution.

1.2 Research Objective and Approach

1.2.1 Intelligent System

The main objective of this project is to develop an intelligent approach to identifying intruder events in a real time scenario. In particular, this project will place emphasis on the higher level intelligent analysis stages of processing and the features required at lower levels to facilitate it. With this in mind it seemed sensible to break the problem up into a set of sequential stages each representing a level of abstraction. Thus the aim is to build a system that will identify intruders based on analysis performed over this sequence of stages, with each stage providing the input to the next.

Primarily this will involve finding a set of models that can represent the behaviour in the real world system at each chosen level of abstraction. The levels explored in this document range from pixel-wise analysis at the lowest level to multiple-frame continuity analysis at the highest. A model of behaviour must be chosen for each level that can allow the development of an algorithm suitable for the constraints of real-time processing. The hierarchy of layers chosen for this work are shown below in Figure 1.1. Further, to aid with justification, each model has been chosen such that an approximation to the probability of event occurrence at that level can be derived. The model at each stage can then be used to validate the viability of a chosen solution by assessing the probability of an event and ultimately by manipulating the model parameters achieve a low probability of failure. Much research has already been done by other authors, covering a vast range of models and techniques applicable to this work. These shall be discussed later in Section 1.3.1 Related Work.

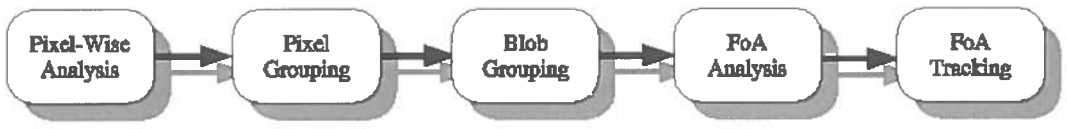


Figure 1.1: Stages of Processing in this System

1.2.2 Design Structure

- Pixel-Wise Analysis

Pixel-wise analysis takes the current frame as input and outputs a binary image (or set of binary images) with white pixels representing those that have been marked as significantly changed in comparison to a reference image (or set of reference images). This system will incorporate two common approaches to selecting a reference image.*Inter-frame differencing*, where the reference image is the preceding frame and *background differencing*, where the reference image is an image of the background. The workings of this stage are described in Section 2.1.

- Pixel Grouping

  The pixel grouping stage of the system receives a binary image of above-threshold pixels from the pixel-wise analysis stage and groups the pixels into blobs of connected pixels. The size of each blob is recorded and only those of significant size are passed onto the next stage. Characteristic properties of each blob (such as mean point and minimum bounding rectangle) are also calculated at this stage. The pixel grouping technique chosen and deductions are fully described in Section 2.3

- Blob Grouping

  The blobs passed in from the previous stage are examined and grouped together based on locality. The new blob group is known as the Focus of Attention (FoA)[1]. Incorporating characteristic information from each blob in the Focus of Attention characteristics preserves them for use at later stages. The foci of attention are then passed to the FoA analysis stage to be processed. The blob grouping process is detailed in Section 3.1

- FoA Analysis

  The characteristics of each FoA are analysed to determine if that FoA is likely to be caused by an intruder. To do this, the FoAs are matched against a set of distinguishing properties used to signify a human shaped object in the frame. These properties have been deduced from empirical and experimental observations. Any intruder FoAs found are subsequently passed to the FoA tracking stage. A full description of the matching process can be found in Section 3.2.

- FoA Tracking

  FoA tracking observes any intruder class FoAs over a sequence of frames determining whether they match the actions of an intruder. Since intruders are deemed to be purposeful actors they will produce certain continuities within a sequence of frames. It is these continuities that must be tracked and categorised to decide if there is an intruder in the sequence. If there are no continuities then processing can continue as normal. Otherwise the system should take the appropriate action to notify of the intruder. The features of the FoA tracking system are outlined in Section 3.3.

An attempt has been made to keep the lower level layers as simple as possible while still providing the necessary data to the later stages. This has been done to both avoid detracting from the main goal of the system (to intelligently detect intruders) and to keep within the bound of real-time computing. The aim of this research is not to find a change detection algorithm (many highly researched techniques exist) but to effectively identify any intruders causing these changes.

In this work **any human activity detected in the sequence** is deemed to be an intruder event. It would be the responsibility of the operator to determine if this is an actual intruder, either by watching the sequence when alarm signalled or by reviewing stored footage. Some new research addresses the difficulties of identifying suspicious behaviour in subjects acting in a video sequence[6]. However determining if such behaviour belongs to an unauthorised perpetrator still remains an unsolved problem. This work has thus made the simplification that any person detected in the sequence is an intruder.

1.2.3 Development

The prototype developed here was developed with the aim to best support an experimental environment. It has consciously been constructed with graphical representations of as much of the underlying computation as possible. This was done primarily in an attempt to facilitate debugging but also had the positive side effect of allowing empirical and heuristic judgments on the best ways to meet the goals of the system.

The system was also developed such that it could be tested initially with a finite set of test data. Using video files as input to the system rather than a live camera feed proved invaluable in identifying faults, and building conjectures on the quality of the system.

1.3 Background

This Section describes the background to the project including work related to the project, the resources available to build the system and the type of data the system will be expected to handle.

1.3.1 Related Work

Lack of references!

Most of the work and research relating to this project can be described by the blanket term Visual Monitoring. This includes work done in scientific monitoring, pattern matching, motion detection, behaviour analysis, surveying, recognition, surveillance and many more. The great deal of work done in all of these areas is mostly driven by the unsuitability of human operators for such tasks. There is such an abundance of research in all these areas that it will be impossible in the time-scale of this project to describe all that could be relevant. For this reason, only the research that is most likely to be relevant to the approach chosen in this project will be considered. The work seen as related to this project can be divided into three categories: change detection, object classification and object tracking.

Within the change detection category the are a huge number of algorithms, which have been developed for a variety of motion processing systems that are adaptable to intruder detection. The change detection techniques are mostly concerned with the low level stages of the system, performing the classification of pixels or small regions as significant or insignificant. The most basic is the simple difference technique which subtracts a reference frame from the current frame to identify changes by the difference of pixels between the images being greater than a set threshold. More advanced statistical variations of this include proximity analysis which lowers the threshold for pixels varying in close proximity. The Shading Model builds on principals developed for 3d computer graphics. It defines the intensity of a pixel from an object in a scene to have a shading coefficient as well as an illumination co-efficient. With this it defines the ratio between intensities in a matching locality. This ratio is then used to fathom whether the shading component of a locality changes and thus if there is some object change.

Object classification is very closely related to pattern recognition. Techniques based on this principal usually operate on higher levels of a system, categorising regions into classes based on their features. Common classification techniques include by geometric features, chromatic features or locality features.

Object tracking relies on having a suitable pattern matching or heuristic technique to build the trajectory information of objects over a sequence of frames. The trajectory of objects can subsequently be further analysed to reveal features of the object's motion. These features can then be used to categorise objects based trsjectory.

1.3.2 Resources

A full video stream producing data at a rate of 18Mbytes/sec would, as well as putting strains on the on the processing and memory resources of a current personal computer, require specialised and expensive digital camera equipment. However, for the purpose of this work a reduced data rate can be tolerated. In this system a web cam producing data at around 3.5 Mbytes/sec is connected through a USB interface to a personal computer installed with the Windows XP operating system. The set-up of these components is shown below in Figure 1.2. The camera captures frames of 320 x 240 pixels using 24bits per pixel to store colour data at a rate of 15frames per second. The personal computer is an x86 architecture machine using an AMD Athlon 1800+ processor and 512MB of RAM.

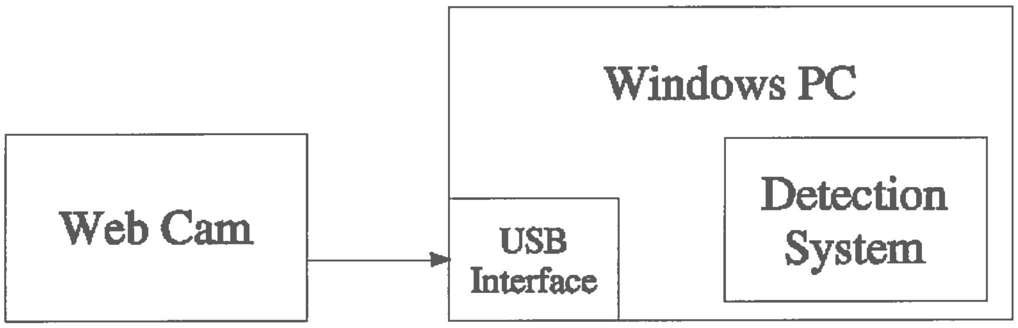


Figure 1.2: Resources Used in This System

2. Low Level Processing

As discussed briefly in Section 1.2 this system will be separated into a sequence of processing stages each responsible for a different layer of abstraction. As shown in Figure 1.1 on page 2 these stages are: pixel-wise analysis, pixel grouping, blob grouping, FoA analysis and FoA tracking. The following Chapter will describe the low-level stages of pixel-wise analysis and pixel grouping and how they were developed to provide the data required by the high level stages.

This Chapter will also discuss two techniques investigated to refine the data output by the frame-differencing algorithm. Specifically these two techniques were investigated in an attempt to eliminate illumination and shadow data from that passed to the later stages of the system. See Section 2.2

2.1 Pixel-Wise Analysis

2.1.1 Overview

The Pixel-wise Analysis (often referred to in academia as chance detection) stage of the system provides a very important function to the later stages. Being the first stage of the detection process it is imperative that this stage provides the maximum amount of significant data to the successive stages. All decisions made later in the system will be made based on data initially provided by this stage so it is paramount that this stage provides what is needed to make those decisions. However, a careful trade off must be made to ensure that the processing required to produce such data for each frame does not take up too much of the timeslot available to process each frame. As the remaining time available in the timeslot will govern the time available for the subsequent stages of processing. The computational load of this stage is expected to be heavy[6] but efforts are made to keep it to a minimum.

One of the most important functions of this stage is the removal of noise from the acquired scene image data. In any image acquired by a digital camera there will be certain amount of error in the pixel representation of a scene. This is unavoidable when attempting to represent the intensity values of continuous universe in a discreet one. However, aside from this quantisation error there is additional error introduced by physical imperfections in the camera's light intensity sensors and their arrangement. This error is particularly high when using a low quality camera such as the web cam used in the project. It is this type of error that causes the majority of pixel noise in an image. Though relatively low in comparison to the scene data represented in an image this noise can cause misclassification when used in change detection.

Another common source of noise is produced by vibration in the camera. Scene pixels in one frame may shift position slightly in the next frame thus causing a mismatch in pixel to pixel correspondences. The difference between these pixel values is then an cause of noise. Such pixels are especially prevalent around object edges. Noise from shake is usually relatively insignificant guaranteed the camera is securely mounted. However in some environments where there are strong vibrations it could cause a more serious problem.

Noise modelling attempts to address the noise issue by constructing a model to represent the amount of noise introduced into an image [13]. Any thresholds used in a change detection algorithm can then be chosen, based on this model, to reduce the misclassification of noise as a significant pixel change. The threshold selection process for this stage of the system is described below in Section 2.1.4.

The choice made for this stage was to implement a statistical approach similar to that first pioneered by Kaup et al[9]. This choice was made because of the comparatively low computational complexity of the statistical technique and also because it is somewhat more tangible than many of the alternative techniques. This tangibility will be particularly useful in the measure and analysis of experimental data.

There are two common approaches to selecting the reference images to be used in change detection. These are inter-frame differencing and background differencing. Inter-frame differencing uses the previous frame as a reference image whereas background differencing uses an acquired background frame[2]. These two techniques each offer a slightly different perspective on the activity in a sequence of frames and thus also the deductions that can be made. This work has used both techniques simultaneously through the early stages of the system in an effort to provide as much data as possible to the later stages.

Often when dealing with noisy image data, frames and reference images are preprocessed with a low pass filter to remove high-frequency spatial variations in the image. These filters can remove much of the noise[11] (particularly that concentrated around edges) in an image but also have a side effect of dulling edges in the image. However, in this project their application was deemed impractical because they are fairly resource intensive and such noisy data will be dealt with in the later stages of processing.

2.1.2 Primary Frame Differencing

This work uses two initial detection techniques to classify between significant pixel events, and random or pixel noise events. Significant pixel events will then be further analysed by the subsequent stages to identify if they are caused by intruder events. These two techniques are referred to in this document as short-term differencing and long-term differencing.

Short term differencing incorporates basic inter-frame differencing between two successive frames in the video sequence. Significant short-term pixel events are defined as those that cause a difference in two successive frames greater than a set threshold. The differences between these frames can be analysed over various channels in a variety of methods. For example the difference may be identified in Intensity, RGB or HSV. The choice of channel will be discussed later.

Long term differencing involves building a background image of the scene and using that image to perform the differencing with the current frame. The background image is constructed by averaging a number of frames not containing any intruders. For example, gradual changes in illumination over the course of the scene's observation should be accounted for by updating the background image as necessary. Starting with the first frame as the background image it is updated when by the next frame not containing any interesting activity. That is a frame, which produces nothing significant after the pixel grouping stage. The updating happens on a pixel by pixel basis using a multiplier α on the old background image and $(1-\alpha)$ on the updating frame to factor each colour value into the new background image. In the prototype system presented here α is set at 0.8. Along with the background image, the standard deviation from it is also calculated. Averaging the $(summed-colour-difference)^2$ between the background image and the image that has just updated it does this(described mathemeticaly in 2.3). The found standard deviation can then from the basis of the dynamic threshold setting described in Subsection 2.1.4.

2.1.3 Differencing Technique

Both the short-term and long-term differencing techniques use a very similar algorithm[9] that effectively subtracts a reference frame image from the current frame image and performs thresholding on the absolute difference of these two images. The intermediate image produced in this process, of the absolute difference, is known as the difference-image and each pixel in it known as a difference-pixel. When this image then thresholded the resulting image is called the threshold-image and it pixels known as threshold or significant pixels.

To take advantage of the colour information supplied by the web cam the algorithm thresholds on the sum of the difference between the red, green and blue components of each pixel. This is an advantage over the traditional monocular differencing techniques, which have trouble differentiating between objects of the same illuminance but different colour. A similar result could be obtained by performing thresholding on each colour channel independently. Though potentially more accurate, when a threshold can be chosen for each channel, it is more costly in memory use. Because of the relatively insignificant loss of accuracy and more efficient implementation the sum of channel differences technique has been chosen. The mathematical representation of the differencing techniques follows below. Colour differencing was implemented originally using the Red, Green and Blue channels because it is the format used in the both the camera and computer representations. Two alternative techniques were experimented with but they were found unsuitable for use in this system. Their implementation and shortfallings are detailed in the next Section, 2.2.

mono channel

2.1.3.1 Short Term Differencing Formula

$$P_{i_c}^{stThresh_{[t+1]}} = \begin{cases} 1 & \left(\sum_{c \epsilon RGB_i} \left(\mid P_{i_c}^{[t+1]} - P_{i_c}^{[t]} \mid \right) > \tau_{ST} \right) \\ 0 & else \end{cases} \tag{2.1}$$

Where $P_{i_c}^{[t+1]}$ is the c th colour value belonging to $\{Red, Green, Blue\}$ of the i th pixel of the current image and$P_{i_c}^{[t]}$ is that of the previous image. τ_{ST} is the short term differencing threshold.

2.1.3.2 Long Term Differencing Formula

$$P_{i_c}^{ltThresh_{[t+1]}} = \begin{cases} 1 & \left(\sum_{c \epsilon RGB_i} \left(\mid P_{i_c}^{[t+1]} - LT_{i_c} \mid \right) > \tau_{LT} \right) \\ 0 & else \end{cases} \tag{2.2}$$

where $P_{i_c}^{[t+1]}$ is the c th colour value belonging to $Red, Green, Blue$ of the i th pixel of the current image $(t+1)$, LT is the Long term background image and τ_{LT} the change threshold for that image.

and where RGB_i represents the individual red, green and blue colour values at pixel .

Background Updating Formula

$$LT^t = \alpha LT^{\overset{t-1}{\boxed{ty}}} + (1-\alpha)new^t \tag{2.3}$$

where LT^t is the background image at time t, LT^{ty} is the previous back ground image, new^t is an image suitable to update the background at time t, and α is the multiplier on the old background image. In prototype system has been set to 0.8 so the background image is effectively an average of 5 frames. | Not Ravg.

Deviation Formula

$$\sigma_{BG}^{2} = \frac{\sum_{i=1}^{N}\left(\sum_{c\epsilon RGB_i}\left(\mid P_{i_c}^{[t+1]} - LT_{i_c} \mid\right)\right)^2}{N} \tag{2.4}$$

where σ_{BG} is the standard deviation of the pixels in frame $t+1$ to those in LT over all N pixels in $t+1$

2.1.4 Setting the Threshold Dynamically

In setting the threshold for use by the two differencing techniques two approaches were investigated. The aim of both techniques is to choose a threshold, which will limit the potential of the system to misclassify noise in the scene as a significant change. The first looks at setting the threshold based on the probability that any pixel difference due to noise shall be above threshold. Whereas the second investigates setting the threshold value such that noisy pixels will produce regions of connected pixels below a certain size. Both threshold-setting techniques derive a threshold based the standard deviation of the pixel noise in an image. Thus they can use the standard deviation calculated with the background image, which is assumed to be an adequate approximation. Furthermore because the standard deviation is being updated along with the background image it will be an accurate approximation at that time. So the threshold will in turn be set dynamically based on the current amount of noise in the scene. This in an advantage over traditional static threshold-setting because the camera noise tends to vary with the illumination conditions in the scene. Dynamic threshold setting means that the threshold now accommodates for changing conditions in the scene.

2.1.4.1 Individual Pixel based Threshold Setting

Following standard statistical approaches ideally the system has a goal of P_{one} for any individual pixel with noise to be misclassified as significantly changed. In this case P_{one} is chosen as $P_{one} = 0.01$. As with many other systems affected by noise the noise in pixel intensity is normally distributed [13]. To achieve an upper bound of P_{one} misclassification we wish to find the threshold value at which probability of misclassification of any individual pixel is P_{one}. Since noise in the pixel intensity is normally distributed around the mean intensity value we can theoretically set the threshold based on this. We want to find a threshold such that the probability the intensity change is out with the threshold is less than $\frac{1}{2}P_{one}$ in both the positive and negative directions. Conversely, considering one half of the normal distribution density function we wish to find a threshold value with the probability that the noise occurs below the threshold is greater than P_one. From the normal distribution table we can lookup a threshold of T standard deviations to give a probability of $\frac{1}{2} - \frac{1}{2}P_{one}$ that the noise is within the threshold. So theoretically T would make a good threshold.

Thus to achieve P_{one}, $\tau_{LT} = 2.58\sigma_{BG}$, where σ_{BG} is the standard deviation from the background image.

Confirmation of this result can be shown by experiment. On generating a guassian noise image of 100 by 100 pixels with a standard deviation of 1, and a mean of 0 it can be seen from Figure 2.1 that selecting a threshold of 2.58 standard deviations results in 106 out of 10000 pixels being mis-classified as significant i.e. Pone=0.0106.

P_{one}

? but σ_{BG} comes from intruders etc. not just noise...

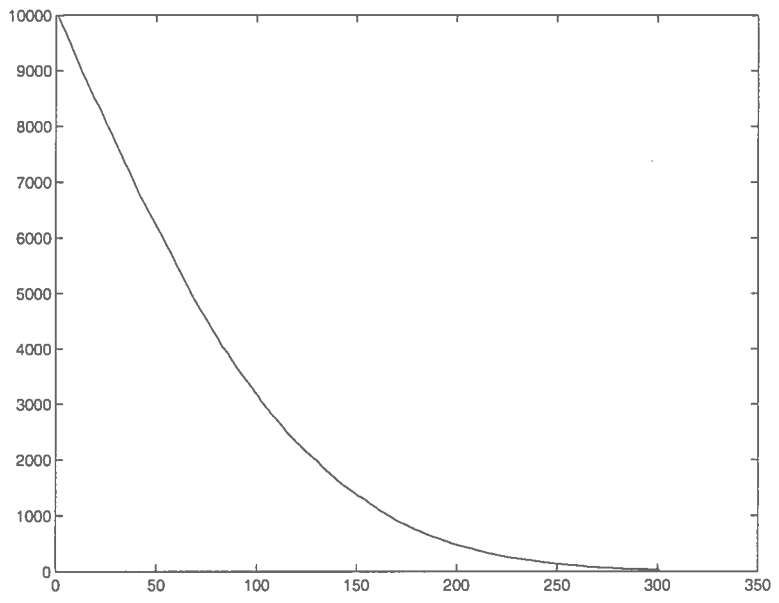


Figure 2.1: Graph of Number of Significant Noise pixels against threshold (x-axis scale $1 = \frac{\sigma}{100}$)

2.1.4.2 Region Size Based Threshold setting

Whilst the last method is effective at setting a threshold which minimises the number of noisy pixel changes misclassified as significant, there is a fair possibility that significant pixel changes are disregarded as noise. However, in reality we do not need to set such a robust threshold because in the next stage of processing we will base decisions of significance on the size of regions of connected significant pixels (blobs). In this case it may be more appropriate to consider a threshold that will keep the number of connected noisy pixels below a chosen size. We are now looking of a threshold that will ensure, while there may be more signifiacnt pixels they will not form groups greater than the size of a blob regarded as significant in the next stage. Choosing this threshold based on the normal distribution of pixel noise is much more of a challenge. To show that the threshold can be lower, we can show that the probability of noisy pixels changing above a threshold being connected is lower than that of any one pixel difference being above the threshold. To see this, consider a 3 by 3 pixel neighbourhood. If the probability of one noisy pixel change above threshold being at centre of the region is Pone then the probability that there are two connected (8 way) pixels in the region is $8P_{one}^2$. This is clearly lower than the probability of $9P_{one}$ that of one significant pixel from noise is in the region.

$$P_{(2connected)} = 8P_{one}^2 \tag{2.5}$$

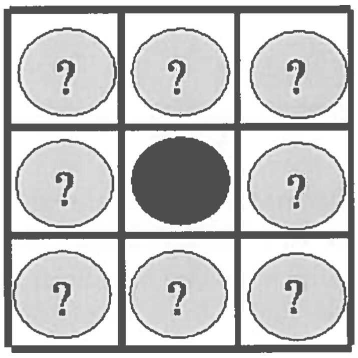


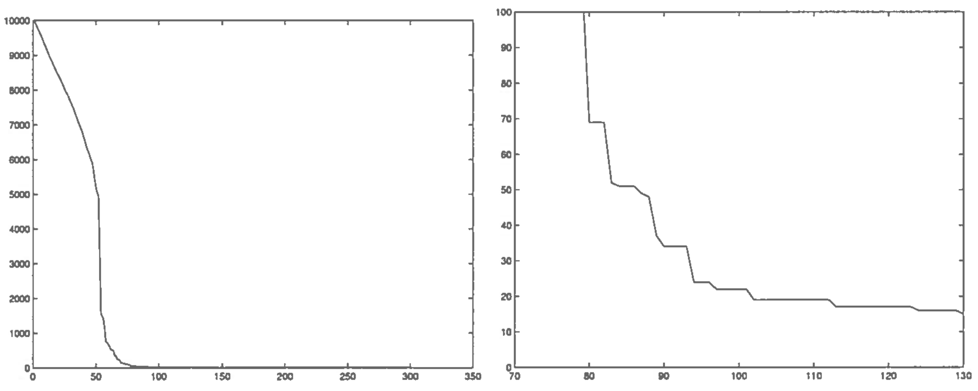
Figure 2.2: Choice in Neighbourhood after First Pixel

However when considering more than two connected pixels the probability derivation becomes very complicated as multiple connections and adjacent neighbourhoods must be considered. This shall not be attempted here (Lieshout[4] has some suggestions). Thus the probabilistic derivation of a threshold to limit blob size cannot be shown here.

Nevertheless, deciding on a threshold that will limit the blob size with respect to noise can be done effectively by experimentation.

Experiment Pseudocode

```
1. Simulated noise by generating a guassian noise image of 100 by
   100 pixels, mean=0, stdev = 1,
2. For thresh = $3\sigma$ to $0\sigma$ stepping $0.01\sigma$
     1 Threshold image with thresh
     2 Group above threshold pixels into regions
     3 MaxRegion[thresh] = largest region size
3. Plot graph of MaxRegion
```



← zoom?

caption single line?

Figure 2.3: Graph of Blob size against threshold (x-axis scale $1 = \frac{\sigma}{100}$)

From this a reasonable approximation of the threshold that will limit region size can be deduced. For example to limit the region size to approximately 30 pixels a threshold of 1 stdev should be set. The size limit on the set region depends on likely size of interesting blobs. Setting the size of interesting blobs is clarified in Section 2.3.

size of image?

From the two algorithms outlined above we can see that the region size limiting approach is more appropriate for this system. Nonetheless care must be taken to set the size limit well below that of blobs considered significant in the next stage. This is emphasised by the fact that the noise produced by the camera is not strictly gaussian. As can been seen from an image of standard deviation in a scene (Figure 2.4) there is a certain structure to this noise. This is due to artefacts caused by JPEG compression used in transmission of images through the USB channel. Part of the compression separates the image into 8 by 8 neighbourhoods thus creating the block structure seen in Figure 2.4. Unfortunately this structure increases the chances that there will be connected regions of pixels. Blobs may be formed running along block boundaries where the noise is greater; shown by the higher (lighter) values along their edges in Figure 2.4.

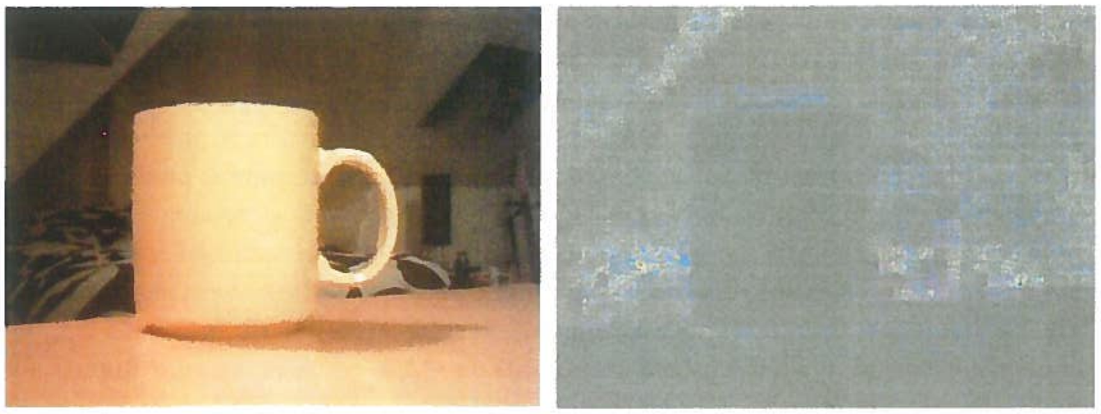


Figure 2.4: Background and Standard Deviation Images from a Scene

Also different objects in the scene tended to produce noise of different levels depending on their illumination, such that the deviation for one object could much higher than that of another in the same frame. From empirical observation it was found that a threshold as high as 9 standard deviations was necessary to suppress region forming pixels when looking for regions of above 30 pixels in the pixel grouping stage. This is far more oppressive than suggested by the two thresholding techniques devised above, making them seem ineffective. Nonetheless their use would be more valid when used with a high quality camera.

2.2 Illumination Invariance

2.2.1 Overview

Varying Illumination in a scene provides several challenges to the positive identification of an intruder. In a simple pixel-wise analysis of the scene, frame differences will be identified because of illumination changes as well as because of changes in the scene. This is because, aside from noise, changes in pixel intensity between frames can be caused by changes in illumination as well as by motion in the scene. These illumination changes will mostly be separate from an intruder event so we wish to consider them separately. Whilst illumination changes may be caused by an intruder, it has been deemed that if the intruder is not in the scene of interest during an event then that event should not be regarded as an intruder event. For this reason we wish to eliminate the consideration of illumination changes as significant events.

Global illumination changes result from an increase or decrease of the ambient light in a scene. In a naturally lit scene global illumination typically changes gradually over the course of a day, and thus is handled as a side effect of the background updating part of the system (described in Section 2.1). Sudden Global illumination changes on the other hand have much greater effect and must be dealt with explicitly. Sudden changes are those which happen within a few seconds. Specifically, those that happen within a time period such that there is a significant difference between the current frame and reference frame. In a naturally lit scene such a change could come about as the result of a fast moving cloud occluding the sun. Alternatively, in an artificially lit scene when lightbulbs are switched on or off. The filtering out of these global illumination changes can either be done in the pixel-wise analysis stage or later by the higher-level processing stages.

Local illumination variance is the change in illumination over particular areas of the scene. The local illumination model occurs as an outcome of the incidence of a point light source on objects in the scene. Any variance in the illumination of the local scene is then caused by an action on those objects or the light source. Possible causes of this illumination change include the creation of a new point light source, its movement or the movement of objects in the scene. The most tangible consequences of interactions in the local model are the casting of shadows.

Local illumination changes as described above can be the product of direct sunlight interacting with objects in the local scene. For example direct sunlight will create brighter areas when it passes through a window onto part of the view. Any object in the path of this sunlight will then cast a shadow in the direction of the sunlight. If at any point the sun is occluded then these localised illumination characteristics will change. Further, these local differences in illumination will cause changes in the intensity values of the pixels in the image that represent the scene. Thus we may have a change detected when there has been no intruder. This is further rationale for the appropriate management of illumination changes in the system.

An additional reason for the pursuit of a system immune to the variation of illumination is due to the complications caused in tracking and categorising an object that casts a shadow. Moving shadows cause problems because they are often mistakingly classified as foreground objects acting in a scene [12]. Moreover they can merge with interesting objects causing a distortion. Thus if caused by an intruder they may distort the features we would ideally like to use to classify intruders. Such a possible distortion of intruder shape can be seen in Figure 2.5. By removing shadows in the scene we could have tighter constraints on the shape properties classifying intruders and consequently more accurate identification. Further the motion of shadows in a two-dimensional representation of a scene is somewhat independent of the casting object. Rather it is dependant on where the shadow casting object is in relation to the light source and the surfaces it casts the shadow on. This varying motion can then cause inaccuracies in the perceived motion of an intruder. As before eliminating shadows would allow the tightening of parameters used to classify the motion of an intruder. Several

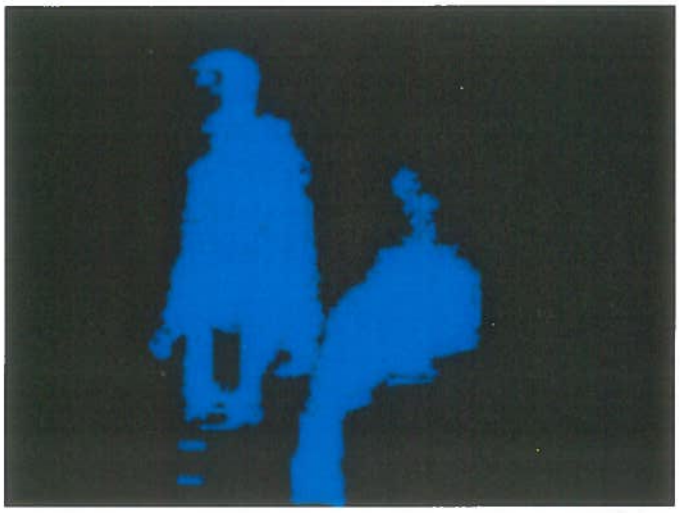


Figure 2.5: Shadow Causing Object Shape Distortion

techniques to cope with varying illumination in a scene have been considered but of those investigated neither were found wholly suitable for the purposes of this project. Conversion from RGB to the HSV colour space and setting constant V gave interesting results though poor camera quality lead to undesired side effects. Homomorphic Filtering was also promising, however high noise variations in low intensity pixels were accentuated making frame comparison inaccurate. The selection of filter shape for homomorphic filtering also raised issues. In the following experiment a gaussian filter was chosen empirically but this may not have given the ideal solution. As recommended by [15] the use of a stochastic filter choice could improve results.

2.2.2 HSV Space - Constant Brightness

The HSV colour space represents colour using three channels; Hue, Saturation and Value. Where hue is what humans would conceptualize as the colour in an artistic sense, Saturation is the purity of the colour (i.e amount of grey, from black to white, mixed in) and Value is the brightness of the colour. In terms of colour data in a scene hue and saturation mostly represent the reflectance properties of objects and value approximates the illumination [14]. These approximations formed a basis to investigate the possibility of illumination invariance by setting value as constant.

The image of a scene acquired from the camera represents the colour data in the standard RGB colour space so this was first converted to the HSV space. The conversion is done using the standard algorithm presented by Foley et al [3]. See Appendix A. Pixel-wise analysis would then be performed over these new RGB values as in the method described in the previous Section. It was hoped that by setting the illumination as constant, thus only considering object reflectance properties, this approach would yield a solution to the problems caused by changing illumination. However, it turned out that keeping value fixed did not set illumination as constant. This can be seen in Figure 2.6, where the shadow behind the subject is still visible. Futhermore in the low intensity pixels the hue data poorly encoded producing artifacts in the resulting image. This was found for two reasons. Firstly, value is only an approximation to illumination, since although hue does not change significantly [12] part of the illumination is also encoded in saturation as well as value. Secondly, noise in the image distorts the colour data spread across the HS and V channels. This is particularly noticable in the artifacts caused by low intensity pixels shown in Figure 2.6. Hence a closer approximation could possibly be obtained from a higher quality camera than that used in the prototype.

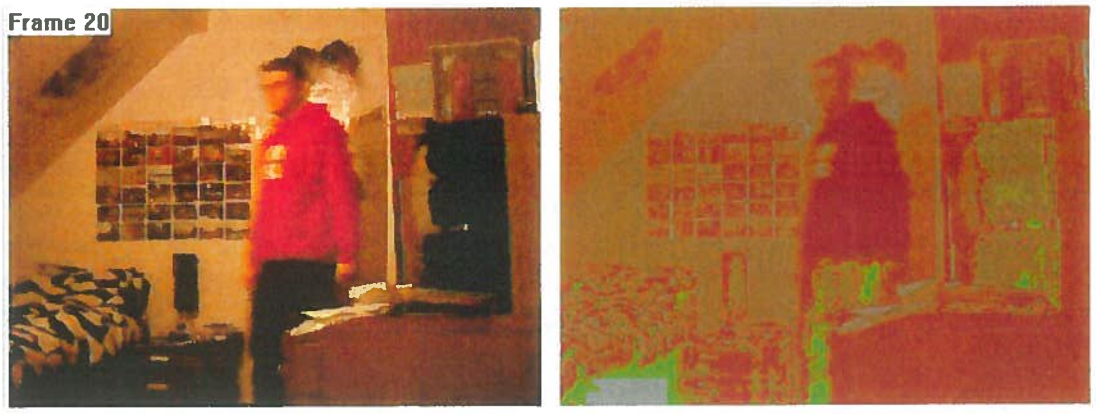


Figure 2.6: Constant Value HSV Conversion

Prati [12] mentions that HSV is an effective tool for shadow identification as shadows cause a more defined drop in saturation than in RGB. This was not implemented in the prototype but would make for interesting future work.

2.2.3 Homomorphic Filtering

Homomorphic filtering is based on modeling the intensity of Lambertian surface objects. That is, objects who's surfaces have the same luminance and radiance in all directions. In such a surface the intensity of each pixel can be expressed as the product of its reflectance and illumination components. Two authors exploit this property to derive homomorphic filtering for illumination invariant change detection. Tot et al [15] developed a technique using a monocular intensity based approach whereas Lou et [10] developed a similar technique using a trichromatic intensity based approach.

$$y_t(k) = i_t(k) \cdot r_t(k) \tag{2.6}$$

Where t is the frame at time t, k is the pixel index, i is the illumination and r is the reflectance component.

This can allow us to separate the reflectance and illumination components of each pixel. By applying a logarithm to formula 2.6 we can convert this relationship to an additive one.

+

$$\log(y_t(k)) = \log(i_t(k) \cdot \log(r_t(k)) \tag{2.7}$$

By assuming that the illumination will be spatially slow varying we can use a low-pass filter over the logarithm of the intensity image to extract the illumination component. This experiment uses a 9 by 9 Gaussian filter matrix with standard deviation set to 2.0.

how done? you don't have i and r separately

$$\log(y_t(k)) = G_{9x9} \bigotimes \log(i_t(k)) \tag{2.8}$$

$$and \qquad r'_t(k) = \exp\Big(\log(y_t(k)) - \log(i_t(k))\Big) \tag{2.9}$$

Subtracting this from the logarithmic intensity image gives the logarithm of the reflectance image. The exponential of thi~~s~~ results in a close approximation of the reflectance image. Henceforth we can use this reflectance image to perform pixel-wise analysis.

In a colour representation of the scene the intensity of each channel is analysed, thus we get equations below.

$$r'_{t_R}(k) = \exp\Big(\log(y_{t_R}(k)) - \log(i_{t_R}(k))\Big) \tag{2.10}$$

$$r'_{t_G}(k) = \exp\Big(\log(y_{t_G}(k)) - \log(i_{t_G}(k))\Big) \tag{2.11}$$

$$r'_{t_B}(k) = \exp\Big(\log(y_{t_B}(k)) - \log(i_{t_B}(k))\Big) \tag{2.12}$$

In spite of offering great potential, when implemented this algorithm did not live up to expectations. The difficulty was in choosing an effective low-pass filter. Toth et al [15] recommend a binomial filter kernel whereas Lou et al recommend a gausian filter kernel. The specifics of the filters size and shape or how to derive them were not revealed. On experimentation it was found that only a mediocre gausian filter could be achieved empirically. Figure 2.7 shows the results of the experimentation. When compared to that of Toth et al our attempt does not provide the same detail in the reflectance image. In fact Toth suggests that a better separation between reflectance and illumination could be obtained by stochastic homomorphic filtering such as used by Fries [7].

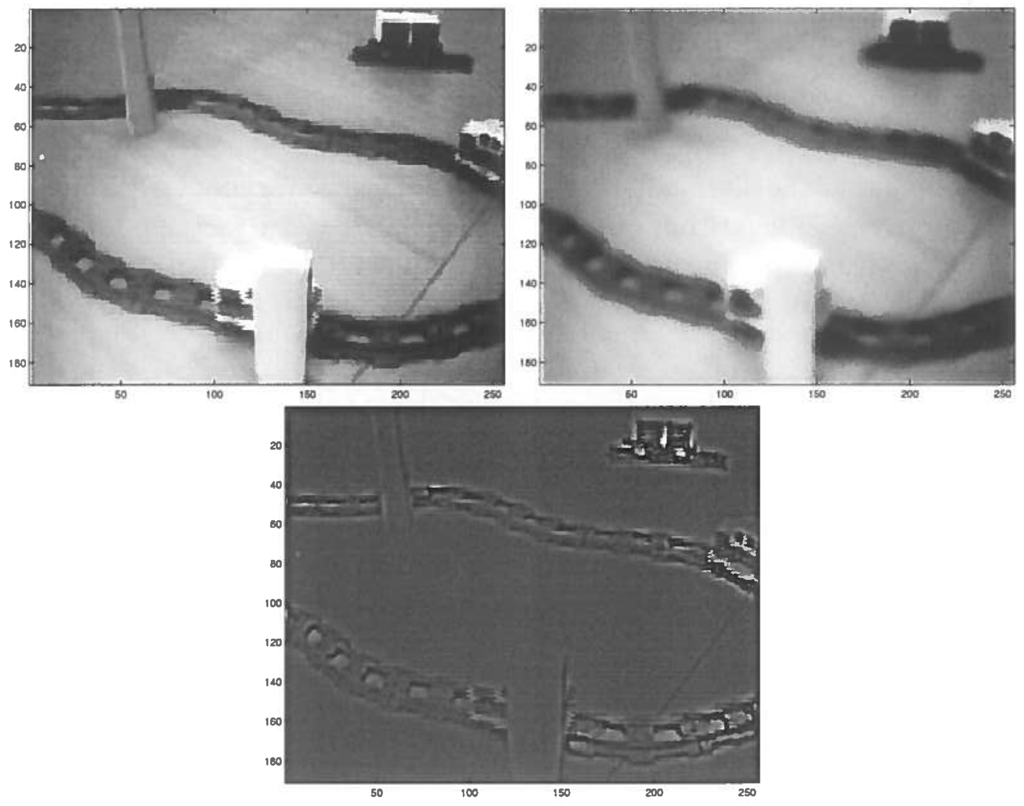


Figure 2.7: Intensity Image, and Illumination and Reflectance Images from Homomorphic Filtering Experiment

Homomorphic filtering was subsequently discounted as an appropriate method for the removal illumination variance. Though could present interesting opportunities for future work if developed using a stochastic filter choosing approach.

2.3 Pixel Grouping

The images received from the previous stage of processing detail the pixels which have significantly changed in both the short and long term. The system must now resolve which of those changed pixels, if any, belong to a larger structure of something interesting in the frame. To find this structure this stage groups the pixels into blobs of connected pixels. As discussed in Section 2.1 groups of significant pixels have a lower probability that they are the result of noise.

The two images received from the previous Section shall be processed independently in this stage as there is no significant gain in information from processing them together.

This system uses a simple 4-way flood-fill algorithm to connect pixels into blobs. In this case a blob is effectively a list of its comprising pixels.

Blob Construction Algorithm

```
1.  push first significant pixel to stack
2.  while there are pixels on stack
3.          pop next pixel P off stack
4.          if (matches found so far is in image of found) continue;
5.          if (pixel P has connected pixel east) push east to stack;
6.          if (pixel P has connected pixel south) push south to stack;
7.          if (pixel P has connected pixel west) push west to stack;
8.          if (pixel P has connected pixel north) push north to stack;
9.          add pixel to blob;
10.         add 1 to size;
11.         update maxes for NorthMax, SouthMax, EastMax and WestMax;
12.         add {X, Y} to {Xsum, Ysum} respectively;
13.         mean coord = Xsum, Ysum divided by size;
14.         get MBR corners from maxes west, north and east, south;
15.         store blob if above threshold size;
```

As shown above, the algorithm also records with each blob the size, mean point, and minimum bounding rectangle of each blob for use in later stages.

As mentioned in Section 2.1.1 there are noise structures in the camera data which could lead to misclassified pixels being connected, and noise from camera shake also produces structured noise. The shift from the shake, as described, will cause noise around edges in the image. Hence in a sequence under shake, significant pixels are likely to be connected along object edges. Further, the width of the significant pixels at an edge depends on the amplitude of the jitter. An example of camera shake is shown in Figure 2.8

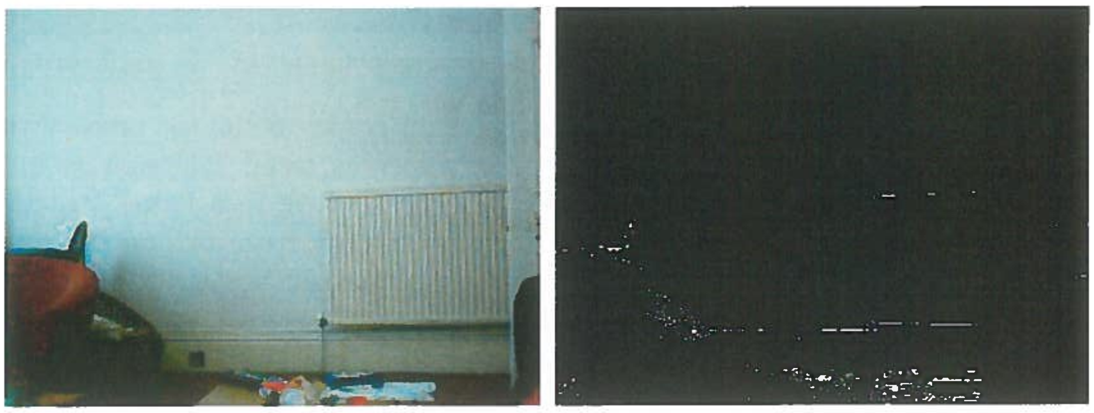


Figure 2.8: Connected Significant Pixels from Jitter

The possibility of structured noise of either source described above means that the size of blobs considered significant must be chosen carefully. Ideally we wish to disregard all those caused by noise. At this stage of the system we are not yet looking to find human shaped blobs. A blob may itself be part of some moving object producing a collection of blobs. The grouping of these blobs is described in the following Chapter. Henceforth, the system cannot yet disregard all those blobs which are of a size smaller than that of a suspected intruder. Consequently we are looking to set a threshold on the minimum size of significant blobs that will exclude those from noise but still include as many from interesting events as possible.

A blob is considered significant if $blob_{size} > \tau_{size}$.

The ceiling on the blob size of pixels from noise was found by experimentation. The experiment involved running the system on a selection of scenes containing only noise or moderate camera jitter and recording the maximum blob size each frame produced. The threshold on pixel difference was set high so as to only regard the pixels which would be caused by some kind of structured noise. A quick scan of the results reveled the the maximum blob size was mostly in the region of two to ten pixels but occasionally there would be a few reaching into the twenties. Thirty was subsequently chosen as a suitable cap.

The algorithm described above constructs the array of significant blobs found in the current frame as shown. This is done for both the short and long term threshold images then the arrays are then passed on to the next stage for the blobs to be considered for grouping.

3. High Level Analysis

This Chapter is concerned with the functions required at a high level to classify the blobs received from the preceding stage into intruder or non-intruder classes. The stages of the system that will be considered in this Chapter (corresponding to the stages in Figure 1.1) are Blob Grouping, FoA Analysis and FoA Tracking. It is these functions that will form the intelligent analysis of the system.

3.1 Blob Grouping

This stage receives the two lists of significant blobs from the pixel grouping stage described in Section 2.3. Like the last, this stage of the system performs the same function independently on the short and long term data. For this system there is little useful deduction that can be made by analysing the relationship between the short and long term blobs. Better analysis can be made later in the system.

Ideally the blobs received would each represent a separate object moving in the scene. In practice however, this is unrealistic as parts of objects will often merge with parts of the background or each other. This leads to some object blobs merging with each other or some being segmented by similarities with the background or . In the short-term case the object itself, in the previous frame, effectively forms part of the background. The blobs in this case are very likely to be segmented because part of the object may have been differenced with itself. Figure 3.1 shows an example of blobs separating where the intruder object in the previous frame has been differenced with part of itself in the current frame.

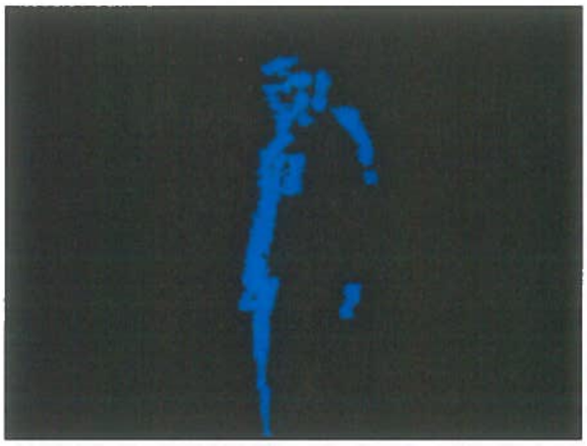


Figure 3.1: Separated Blobs Belonging to One Object

The goal of grouping is to take a list of blobs and find where blobs in the list belong to objects consisting of a group of them. When such a group of blobs is found they are refactored into a new structure know as the Focus of Attention (FoA) (alternatively sometimes called a Region of Interest). Blobs which are found not to belong to another group, become themselves the Focus of Attention. The aim of this stage is thus to have each moving object represented by a unique Focus of Attention. Again, for similar reasons as the last case, this is a slightly unrealistic goal. Nevertheless, because the application of this system is intruder detection we expect only to be dealing with a small number of moving objects. We are almost certainly expecting the number of intruders found in a scene to be smaller than ten and in the most likely instance to be only one or two. It is thus most likely that separated blobs within a locality are part of an intruder. This means that having an FoA construction algorithm with a propensity towards grouping blobs rather than separating them will provide an effective solution.

FoA construction can be done based on a variety of qualifiers which match separated blobs to a group. For the purposes of this work a heuristic locality method was selected. This is based on the princip~~al~~le that blobs in a locality are likely to be part of the same intruder. It would also be worth using some sort of weighting on the size of blobs to govern the extent of locality considered. The intersection of blob Minimum Bounding Rectangles (MBRs) provides a fair approximation to this locality and size weighting. It works similarly to the technique of Bruce[5], but without using the density factor. This system uses the density factor later in FoA Analysis (Section 3.2. Empirical observation of MBR blob matching showed the MBR intersection to work well grouping blobs belonging to the same object. It may not capture every blob that belongs to an object but it will group a high enough proportion of them to get reasonable results from the FoA , analysis stage of the next Section.

The algorithm compares the extremities of the MBRs for each blob in the scene and groups those with intersecting MBRs. The implementation (shown in the algorithm below) compares every blob to every other, grouping them if there is an intersection. It then reiterates over the blob list with the new grouped-blob (Temporary FoA) replacing those it grouped with itself. This replacement is possible because blobs and FoAs are described using the same vector of properties. Eventually when there are no more intersecting blobs the algorithm completes.

FoA Construction Algorithm

```
1.  for each blob i in the scene list;
2.      for each blob j in the scene list;
3.         if MBRs of i and j intersect;
4.            construct FoA f of i and j pixels;
5.            set new MBR of f as greatest extremities;
6.            set centroid of f ((i.centroid * i.size)
              +(j.centroid * j.mean))/(i.size+j.size);
7.           set size of f as (i.size*j.size);
8.           remove blob i and j;
9.           add new FoA f to scene list;
10.          continue;
11.  return new list of FoAs;
```

The algorithm also combines the MBR, size and centroid features every time it amalgamates two blobs, preserving them for analysis by subsequent stages. It should be noted that when blob size is compounded into an FoA it becomes a measure of active connected pixels.

Using this algorithm, blob grouping is performed on both the long-term and-short-term blobs. The resulting FoA arrays are then passed to the next stage.

A slightly more efficient implementation could be achieved using an iterative scan-line algorithm. The instead of iterating when a pair of intersect blobs is found it would find all immediately intersecting blobs then iterate on the new blobs formed from these. However, the gain in efficiency would not be dramatic because we are only concerned with a relatively small number of blobs in the scene.

One possible drawback of using an MBR based grouping function the tenancy to include shadows in FoAs. Shadow blobs are usually close enough to the subject that the will become part of the FoA. Ideally they should be ignored, however this system does not differentiate between object blobs and shadow blobs so they must be tolerated. This will cause difficulties for the subject classification part of the FoS analysis stage (Section 3.2) On the other hand having them in a separate FoA could cause tracking errors later in the system.

3.2 FoA analysis

The purpose of the FoA Analysis stage is to take the two arrays of FoAs provided by the previous stage and decided if any of the FoAs they contain match to the static features of an intruder. The features available in the FoAs for classification come from its shape, size and active pixel density.

However, due to the inherently high blob segmentation of objects in short-term FoAs, they are not suitable for use in a feature classifier. As mentioned in Section 3.1 this is caused by the performance of the frame differencing being between parts of the same object. The long-term FoAs provide a more appropriate depiction of the intruder because they more clearly represent the complete shape of a potential intruder. The reason for this is that the background subtraction method of Section 2.1 gives a strong segmentation between the background of the scene and objects pixels acting in the foreground. Of course there are still issues of occlusion, object-background blending and object merging but the feature recognition system can be set with robust parameters to factor for them.

One of the more prominent issues for feature classification is the large distortion of object shape caused by the misclassification of shadows as acting foreground objects in a scene (Section 2.2). Shadows can even account for an area of active pixels greater than that of the target object. Fortunately there are a few properties of shadows that can be exploited to allow classification of intruder shapes despite their presence. Shadows under normal overhead lighting conditions will almost always fall below the subject. That is, they will not appear in the image extending above the height of the subject. Also, again under normal overhead lighting, the magnitude of their extension in the x component of an image will not usually exceed the height of the subject.

These two features can then be used to bound the expected dimension ratios of the of a subject. Under the shadow constraints above, the width of the MBR of a subject and shadow combined will not be expected to exceed the height. Since the next stage will further analyse the features of subject movement before classify between intruders and non-intruders we can afford set such a generous threshold.

$$MBR_{width} < MBR_{height} \tag{3.1}$$

a ratio threshold can thus be set as

$$\frac{MBR_{height}}{MBR_{width}} < 1 \tag{3.2}$$

Putting a lower bound on the dimensions of the MBR also requires generous threshold setting because aside from occlusion and other shape distorting factors the subject may appear at any distance from the camera. In this system the limit on minimum MBR dimensions has been set very low.

$$MBR_{width} > 20_{pixels}\ and\ MBR_{height} > 30_{pixels} \tag{3.3}$$

In a closed environment, where the furthest subject distance from the camera is known, it is worth noting that we can set this threshold very specifically to the minimum expected person size at that distance.

The last feature this system shall consider is the density of active pixels found in the FoA's MBR. Since the subject we are aiming to classify is a the FoA of a long-term frame difference, and presuming the thresholds in the previous stages have been set adequately sensitive, we shall be expecting the active pixels to be covering a significant area of the MBR. From experimentation it was found that the this coverage ratio (or pixel density) of the MBR seldom dipped below 1/5.

$$\frac{FoA_{activepixels}}{(MBR_{height} * MBR_{width})} > frac15 \quad \frac{1}{5} \tag{3.4}$$

The general algorithm to perform tests for the three constraints described above iterates through the FoAs in the long-term array to find one which matches constraints. If an FoA matches the constraints; pass both FoA arrays to the FoA tracking stage.

On the other hand if there are no intruder shaped FoAs found in the current frame then it should be used to update the background image as described in the background updating formula 2.3 ✓

3.3 FoA Tracking

This stage of the system provides the tracking and motion analysis over a sequence of frames for the FoAs found in the short-term and long-term difference frames. The output of this stage is the record of frames in which intruder activity has been recorded. These intruder events are classified on the premise that an intruder will be a purposeful actor over a sequence of frames.

When this stage first receives non empty arrays from the previous stage (i.e. there has been a shape classified as intruder shaped.) it will enter the intruder_finding state in which it considers a sequence of 25 frames. An intruder can only be detected if the system matches the features of intruder motion over these 25 frames. 25 frames was selected an appropriate number of frames to be considered as it is expected that any purposeful motion of an intruder will last at least 1.3 seconds ($\frac{25_{frames}}{15_{frames/sec}}$) . Once an intruder has been detected it then continues in intruder_found state. In this case it will continue over blocks of 200 frames, restarting in the intruder_finding state after that.

The FoA tracking algorithm implemented for the purposes of this system is very simple. It makes the assumption that there will only be one intruder in the scene or if more than one intruder they will be moving along the same path as each other. This generalisation allows the intruder or group of intruders to be tracked based on the centroid of all the FoAs contained in each array. The tracking part of this stage is performed independently on the short term and long term arrays. They will be combined later, in the analysis part of this stage to formulate decisions on intruder detection. FoAs are tracked by storing the centroid for each of the 25 frames in the intruder_finding state. From this the vectors between the centroid points are calculated. Also during the storing, the maximum separation from the first recorded centroid of the sequence is updated. A better algorithm which will track multiple FoSs as they merge or split over a sequence of frames is discussed in the following Subsection. This algorithm, though not implemented in this system, would make a very considerable improvement to future work in this area.

Shadows as described in Section 2.2 may also cause problems for this stage of the system. However they will usually not be of an area greater than that of the subject so the centroid will not shift drastically. Also they will follow the subject (if little erratically) so motion properties used to classify intruders in the next step of processing will not be greatly affected.

The are a variety alternative methods to track objects rather than just selecting the centroid of the FoAs. These include MBR intersection, colour histogram matching and other pattern matching techniques.. The system would be able to perform intruder tracking on multiple targets and thus more robust detection.

3.3.0.1 A better Tracking Algorithm

Fuentes[8] proposes a *blob*[1] tracking Algorithm that can cope with the merging and separating of blobs. They use a matching matrix to form a bidirectional matching function between two consecutive frames. The matching matrices are constructed with the old blobs $B_j(t-1)$ matched against the new blobs $B_j(t)$ forming M_{t-1}^{t} and the new blobs against the old forming M_t^{t-1}, as shown in 3.5

$$M_{t-1}^{t}(i,j) = \text{Matching}\big\{B_i(t-1), B_j(t)\big\} \quad (3.5)$$
$$M_{t}^{t-1}(i,j) = \text{Matching}\big\{B_i(t), B_j(t-1)\big\} \quad (3.6)$$

To follow the matching they also use a *matching string* of the form:

$$S_{t-1}^{t}(i) = \bigcup_j j \text{ such that } M_{t-1}^{t}(i,j) = 1 \quad (3.7)$$

This records the blob number of each blob in the considered frame that matches with each in the blob other frame done for both the matching directions. An example of the formed matrices follows.

[1]Fuentes definition of a blob is what has been defined in this document as an FoA

$$M_{t-1}^{t} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.2:

$$M_{t}^{t-1} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.3:

$$S_{t-1}^{t} = \begin{bmatrix} 1 & 1 & {}^{2}_{3} & 4 \end{bmatrix}$$

Figure 3.4:

$$S_{t}^{t-1} = \begin{bmatrix} {}^{1}_{2} & 3 & 3 & 4 \end{bmatrix}$$

Figure 3.5:

To perform the matching of blobs between frames Fuentes uses a blob MBR intersecting function basing it on the principal that blobs are not likely to move greater than the spacial extents of it's MBR. However this could be extended with a more to be more robust, using techniques such as histogram matching, quartile centroid matching. Like the prototype system the trajectory is calculated by the centroid difference vectors. When blobs merge the the data of the centroids for both is store and a new group made encompassing the two blobs. When the group splits the centroid data, and thus the trajectory, of the stored blobs is restored.

3.3.1 Motion Analysis

The motion analysis of this stage analyses the data stored for 25 frames when in the intruder_finding state or for 200 frames if in the intruder_found state. It uses both the short term and long term FoA arrays to decide if the trajectory data found in the previous tracking part corresponds to that of an intruder.

In this stage of the system the most important function is to differentiate between movement belonging to some object that is roughly intruder shaped but not an intruder and that of an intruder. The most common of cause of such motion will be environmental fluctuations caused by the wind, particularly plant life. Animals should have been factored out in the previous stage because they will not match the general shape of an intruder defined.

From empirical observation and basic experimentation it was concluded that the clearest difference between intruders and plant life was found to be the furthest separation from the starting centroid of the motion. Plants would not be expected to move very far from the start of their motion. Whereas a purposeful intruder would likely traverse a significant distance in the frame.

From experimentation on a variety of scenarios in which plant life is moving it was found that only in extreme cases plants moved a distance over thirty pixels from the start point of their motion. A threshold of 60 pixels was thus set to incorporate the short term and long term trajectory information. (i.e.$st_{maxsep} + lt_{maxsep} < 60_{pixles}$)

Resetting the system every 200 frames is done to prevent a problem that was encountered when a subject entered a scene with moving vegetation in the background. After the intruder left the scene the system falsely assumed that the continuing motion of the vegetation was that of an intruder.

Sept 4
Fr, [illegible]

4. Evaluation

The goals outlined in the introduction to this document state that an intruder detection system should be able to positively identify intruders with the lowest possible false alarm rate. This chapter will evaluate how well the system matches this criteria in two way. Firstly it shall discuss how the system will cope in specific scenarios, outlining what is provided in each stage of processing to deal with the scenario. Secondly it shall evaluate the system based on how it performs under testing.

4.1 Detection Scenarios

This section will outline how? There are four main scenarios that the intruder detection system will have to cope with. They shall be discussed in turn, detailing how the system presented in this paper shall react.

1. No intruder in the scene with no movement in the background.

   In this scenario the system will discount the possibility of intruder presence in the early stages of processing. The pixel wise analysis stage will factor out the majority of pixels changing due to noise. The remaining pixels will be passed to the the pixel grouping stage, which will group any that are connected. These will always almost never form groups of any significant size so shall be removed. The main challenge here is if the threshold images from the pixel-wise analysis stage contain regions of connected pixels caused by camera vibration. However the pixel grouping stage will quickly discount them as it has been designed specifically to factor out such regions. In the unlikely event that the regions are of a greater size than the threshold set by the blob grouping stage they shall certainly be removed by the FoA analysis stage.

2. No intruder in the scene but with some illumination change

   Gradual illumination changes in the system will be effectively dealt with by the background updating function of the system. However, sudden illumination changes may pose more of a threat. Under normal circumstances such changes will be either in the whole scene or within one particular region. It is possible they can cause a change that would match the features of an intruder in the first four stages of the system. However, they will not cause any motion so will certainly be removed by the FoA tracking stage.

3. No intruder in the scene but with some sort of movement in the background

This is the most complicated non intruder scenario to deal with because it is possible that the object causing such motion will match the features used in the first four stages of the system to identify intruders. The most likely cause of such motion in a scene will be from vegetation moving in the background. For this reason the system has been specifically designed to classify the motion of such motion as not that of an intruder. Thus the system will effectively discount any vegetation moving in the background. Animals though capable of following trajectories similar to that of an intruder should be removed by the FoA analysis section as their shape features are somewhat different from that of an intruder.

4. An intruder in the scene with any combination of above activity in the background

   An intruder in the scene will cause a significant number of connected pixels to change above threshold in the pixel-wise analysis stage. The size of the region of these pixels will be greater than that of the pixel grouping threshold so the blobs they form will be passed to the blob grouping stage. Any blobs in close proximity shall then be grouped into an FoA. The intruder FoAs will match the features used to classify them in the FoA analysis stage and be passed onto the FoA tracking stage. FoA tracking will find the motion of the FoAs to be that of an intruder and subsequently signal an intruder has been detected.

4.2 Testing

To evaluate the system testing this paper will a Receiver Operator Characteristic curve (ROC curve)[1]. This technique has been chosen because it will give a statistical measure of the competence of the system and allow observation of the effect of changing the threshold parameters. An ROC curve plots the true acceptance rate of a system against its false acceptance rate. In this system that is the rate at which intruders are identified correctly against the rate non-intruders are classified incorrectly as intruders. An example of the curve produced by an ROC is shown in Figure 4.1

The system has been tested under a variety of real scenarios using the video files described in Appendix B . There are thirty test files in total each used to test a slightly different functionality of the system. To gain the acceptance rates needed to produce an ROC curve the test data has to be tested in the system over a varying threshold [1]. The threshold chosen to vary is that of pixel differencing in the Pixel-Wise Analysis stage. This threshold has been chosen because it will test the competence of the later stages when confronted with a

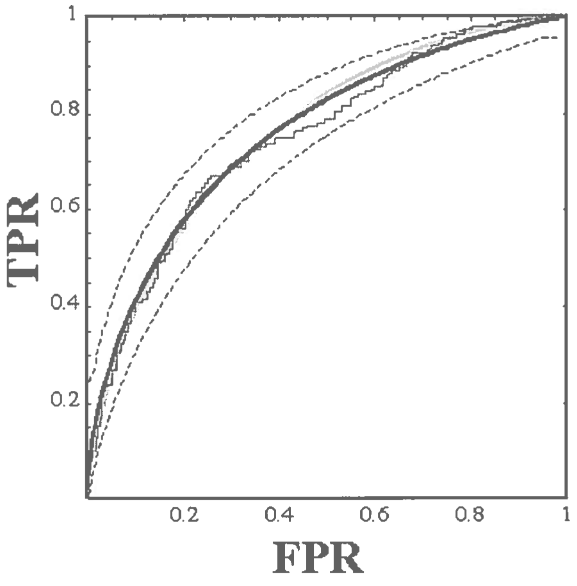


Figure 4.1: Example ROC Curve

very large or very small amounts of significant pixel data. To produce the ROC curve for this system each file was input to the test system, which iterated over 350 pixel difference threshold values [1] performing intruder detection at each one. The frames numbers in which an intruder was found were then stored for further analysis. Each test sequence was also observed by a human operator to get the ground truth values for intruder frames. For each threshold in a test where the system detection matched the ground truth detection a true positive result was recorded, and where there was a mismatch a false positive was recorded. At each threshold the sum of the true positive and false positive were summed over the thirty test sequences. This gives us the true acceptance rate and false acceptance rate for each threshold. The true acceptance rates and false acceptance rates were then plotted against each other to obtain the ROC curve in Figure 4.2.

This curve is is very jagged because of the small sample size of the test data. Ideally the analysis would have been performed over hundreds or thousands of test sequences. However, it still shows some interesting results. The reason the lower leg slopes across the graph, instead of downwards as in the example ROC curve, is because when the system fails to identify intruders it will more likely fail to falsely identify them aswell. This happens because of the classification features of the later stages are robust to the features of an intruder.

The quality of a system producing an ROC curve is usually judged by closest point

[1]Note the dynamic threshold setting described in Section 2.1 was disabled to perform these tests

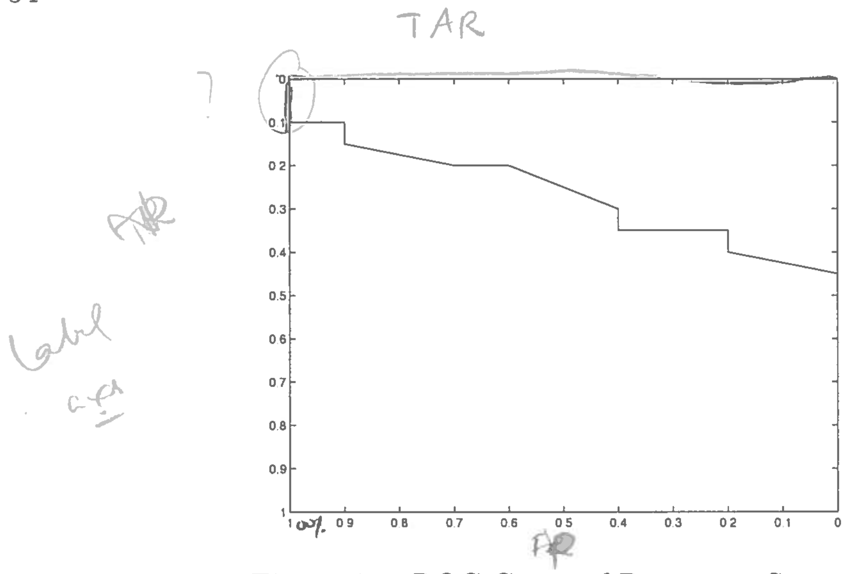


Figure 4.2: ROC Curve of Prototype System

of the curve to the corner of 100% true acceptance rate and 0% false acceptance rate. The graph in this case actually claims that the system achieves this. In fact it claims a zero false acceptance rate (flush against top x axis) for a very large proportion of the system.

It could be deduced from these results that the intruder detection system presented here is almost perfect, when the right threshold is chosen. However that is not necessarily true. For one, the sample size of the test data is very small. Further there is a bias in the results because the test data used is that which was used to test the system during development.

In spite this it is still a very positive result.

4.3 Conclusion

With such a vast range of possible intruder characteristics general intruder detection is a very difficult problem. The system presented here has made a good attempt to draw on some of the defining characteristics of intruder action in a video sequence and use them to classify an intruder. The analysis of the first Section of this Chapter show that the system should be able to deal with any eventuality, though there is some possibility for inaccuracy. The results from the previous Section show that the system works very well on the test data used but needs much more thorough testing.

The prototype system also runs well within the bounds of real time computing. Though the frame rate of the camera used for input is only 15 frames per second the video files used in testing were being processed at around 25 frames per second.

Additional, features could be added at every stage to make the system more robust. However, adding such features will increase requirements on resources. As it stands, the system performs well in real time so any further improvements must be carefully considered.

4.4 Future Work

Much of the future work for this system has discussed through this document including that of shadow and illumination removal (Section 2.2) and multiple blob tracking (Section 3.3).

Also a suitable intruder alarm handling function could be implemented. At the moment the system does very little apart form record when an intruder has be identified. The next stage of the system would be to implement a sequence recording system. The Win32 API provides very simple functions that could be used to compress and store *avi* frame sequences to video files such as those used in testing.

Appendix A. HSV Conversion Algorithm

```
void RGB_To_HSV(double r, double g, double b, double *h, double *s, double *v)
/* Given r,g,b each in [0,1] */
/* Desired: h in [0,360), s and v in [0, 1] except if s=0, then h = undefined, */
/* which is some oncstant defined with a value outside the interval [0,360]. */
{
        double max = maxi(r,g,b);
        double min = mini(r,g,b);
        *v = max;
        /* Next calculate satuation s, Satuaration is 0 if red, green and blue are all
        *s=(max!=0.0)?((max-min)/max):0.0;
        if (*s==0.0)
                *h = UNDEFINED;
        else {
                double delta = max-min;
                if (r==max)
                        *h=(g-b)/delta;
                else if (g==max)
                        *h=2.0+(b-r)/delta;
                else if (b==max)
                        *h=4.0+(r-g)/delta;
                *h*=60.0;
                if(*h<0.0)
                        *h+=360;
        } /*chromatic case*/
} /*RGB_To_HSV*/


void HSV_To_RGB(double *r, double *g, double *b, double h, double s, double v)
/* Given h in [0,360] or UNDEFINED, s and v in [0,1]. */
/* Desired: r, g, b each in [0,1] */
{
        if(s==0.0)  /* The colour is on the black and white centre line. */
        {
                /* Achromatic colour; there is no hue. */
                if(h==UNDEFINED)
                {
                        *r=v;
                        *g=v;
                        *b=v;
                }
                else
                {
                        wsprintf(report_buf, "BAD HSV!");
                        //MessageBox( app_win, report_buf, "bad", MB_OK);
                        *r=v;
                        *g=v;
                        *b=v;
                }
        }
        else
        {
                double f,p,q,t;
                int i;

                if (h==360.0) h = 0.0;
                h/=60.0;
                i = (int) floor(h);
                f=h-i;
                p=v*(1.0-s);
                q=v*(1.0-(s *f));
                t=v* (1.0 -(s*(1.0-f)));
                switch(i)
                {
                        case 0: *r=v; *g=t; *b=p; break;
                        case 1: *r=q; *g=v; *b=p; break;
                        case 2: *r=p; *g=v; *b=t; break;
                        case 3: *r=p; *g=q; *b=v; break;
                        case 4: *r=t; *g=p; *b=v; break;
                        case 5: *r=v; *g=p; *b=q; break;
                }
        }/*chromatic case */
}/* HSV_To_RGB */
```

Appendix B. Test Data File Classification

- File 1 Intruder walking through a cluttered badly lit scene
- File 2 Intruder, sudden entry and exit
- File 3 Intruder walking towards camera
- File 4 Intruder walking slowly through scene, pausing, then leaving
- File 5 Two intruders entering and leaving scene together
- File 6 Intruder entering and leaving crouched
- File 7 Intruder walking quickly through scene
- File 8 Intruder moving erratically through scene
- File 9 Intruder moving through scene with moving plant in background
- File 10 Intruder walking through scene with illumination change.
- File 11 Small plant with moving leaves
- File 12 Small plant swaying gently
- File 13 Small plant swaying vigorously
- File 14 Large plant with moving leaves
- File 15 Large plant swaying gently
- File 16 Large plant swaying vigorously
- File 17 Two plants with moving leaves
- File 18 Two plants swaying gently
- File 19 Two plants swaying vigorously
- File 20 Plant swaying gently with illumination change
- File 21 No movement, cluttered scene in good lighting
- File 22 No movement, uncluttered scene in good lighting
- File 23 No movement, objects in close proximity
- File 24 No movement, cluttered scene in poor lighting

- File 25 No movement, uncluttered scene in poor lighting
- File 26 No movement, illumination change (dark to light)
- File 27 No movement, illumination change (light to dark)
- File 28 No movement, camera vibration, cluttered scene
- File 29 No movement, camera vibration, uncluttered scene
- File 30 No movement, camera displacement

Bibliography

[1]

[2] *Computer Vision.* Blackwell, 1988. Auth?

Auth [3] *Computer Graphics–Principles and Practice.* Addison Wesley, 1990.

Auth [4] *Highly Structured Stochastic Systems*, chapter Probabilistic Image Modelling. European Science Foundation, 2003.

[5] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[6] G.L Foresti and C.S. Regazzoni. New trends in video comunications, processing and understanding in surveillance applications. In *ICIP.* year

[7] R.W. Fries and J. W. Modestino. Image enhancement by stochastic homomorphic filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1979.

[8] L.M. Fuentes and S.A. Velastin. People tracking in surveillance applications. In *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance.* year

[9] T. Kaup, T. Aach, and R. Mester. Statistical model-based change detection in moving video. *Signal Processing*, 1993.

[10] J. Lou, H. Yang, W. Hu, and T. Tan. An illumination invariant change detection algoritm. In *The 5th Asian Conference on Computer Vision.* year

[11] J. Nascimento. Intelligent intruder detection from a video sequence. where, year?

[12] R. Prati, A.and Cucchiara, I. Mikic, and M.M. Trivedi. Analysis and detection of shadows in video streams: A comparative evaluation. *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 2001.

[13] P. Rosin. Thresholding for change detection. *British Machine Vision Conference*, 1997.

[14] William Shoaff. Color, illumination models, and shading. When?

[15] T. Toth, D. Aach and V. Metzier. Illumination-invariant change detection. In *IEEE Southwest Symposium on Image Analysis and Interpretation.*
year