Using computer vision techniques to count and track humans in surveillance video.

Scott Tattersall

B.A. (mod) Computer Science
Final Year Project May 2002
Supervisor: Dr Kenneth Dawson-Howe

Acknowledgements

I must thank my parents for all their help and for their painstaking proofreading of this project, my flatmates for their understanding, and my girlfriend for all her support throughout. Finally, I have to thank my project supervisor, Dr Kenneth Dawson-Howe, for all his time, help and direction throughout this project's development.

Table Of Contents

Acknowledgements ..... **ii**

Report Structure ..... **vi**

CHAPTER 1 – INTRODUCTION **1**

1.1) Purpose: **1**

1.2) Aims: **1**
1.2.1) Initial Goals: 1
1.2.2) Motivation: 2

CHAPTER 2 – BACKGROUND INFORMATION **3**

2.1) What we have: **3**
2.1.1) General Background Information: 3

Specific Background on relevant Techniques: **6**
2.1.2) Optical Flow analysis: 6
2.1.3) The Kalman Filter: 7
2.1.4) Neural Networks: 9
2.1.5) Skin Colour analysis (in HSV Space): 11

2.2) What we need **12**

CHAPTER 3 – PLAN & DESIGN **13**

3.1) Introduction **13**

3.2) Software development lifecycle **13**

3.3) Stages of development **13**

3.4) – Overall Algorithm Definition **16**
3.4.1 – Background subtraction: 16
3.4.2 – Noise Reduction (Convolution) 17
3.4.3 – Binary Morphology (Opening & Closing) 19
3.4.4 – Connected Component Analysis 20
3.4.5 – Basic Tracking 22
3.4.6 – Grouping People 23
3.4.7 – Generating Colour Histograms 25
3.4.8 – Using Histograms to aid tracking 27
3.4.9 – The Adaptive background model 28
3.4.10 – The Estimation of group size 29
3.4.11 – The problem of Shadows 30

3.4.12 – Dealing with occlusion (Proposed solution) 36
3.4.13 – The sample application 37

3.5 – Summary 38

CHAPTER 4 – ACTION & IMPLEMENTION 39

4.1) Introduction 39

4.2) – Implementation of algorithm stages 40
4.2.1 – Finding Moving Pixels and the adaptive background model 40
4.2.2 – Noise Reduction 41
4.2.3 – Connected Component Analysis 42
4.2.4 – Basic Tracking 44
4.2.5 – Grouping People 45
4.2.6 – Generating Colour Histograms 47
4.2.7 – Using Histograms to aid tracking 48
4.2.8 – Estimation of group size 50
4.2.9 – Shadow removal 51
4.2.10 – Sample Application 53

4.3) Classes and Data structures 55

4.4) Summary 56

CHAPTER 5 – REVIEW 57

5.1) What Went Well & Why 57

5.2) What could be improved & how 57
Algorithm additions: 58
5.2.1 – Complex backgrounds 58
5.2.2 – Rapidly changing illumination 60
5.2.3 – Large group size estimation 61
Functionality enhancements: 61
5.3.4 – Additional object information 61

CHAPTER 6: SAMPLE RESULTS 62

CHAPTER 7: REFERENCES 63

APPENDIX A – "SHADOW REMOVAL METHOD B ILLUSTRATED" 66

APPENDIX B – "SOME VISION CONCEPTS EXPLAINED" 69

Colour spaces: 69
RGB: 69
HSV: 70

Terms & Concepts: **71**
Occlusion: 71
Quantisation: 71

Report Structure

The structure of this report reflects the planning method used for the project. That is the Coverdale Systematic Approach (CSA) to Task Completion. It was felt necessary to use some method of formal design and time management in order to complete the project properly. CSA is a way of structuring a Job to maximise efficient and relevant development as is basically as follows:

1. Purpose	Why is the project being created?	*Preparation*
2. Aims	The goals of the project, Provides a measure of success	
3. Information	What we have	
	What we need	
4. Plan	How are we going to achieve the goals?	
5. Action	The Implementation of the Plan	
6. Review	What Went Well & Why?	
	What Could be Improved & How?	

It is a proven method of tackling a problem effectively without spending too much time with superfluous planning (paralysis by analysis) [**Coverdale**].

Chapter 1 – Introduction

1.1) Purpose:

The basis of this project is to apply computer vision techniques to enable intelligent surveillance video processing. That is to have a computer "watch" a surveillance video and provide useful information its content. The main goal of the project is to count the number of people that pass in front of the camera (in any direction) during a given time period.

1.2) Aims:

1.2.1) Initial Goals:

The main goals of this project are:

1) Tracking of moving objects on the surveillance video.
2) Counting the number of humans passing through the recorded scene.

The possible sub-goals for the project are:

- Storing descriptions and still images of the individual people passing in front the camera.
- Allow searching of video sequence based on key features (such as hair colour, sex and clothing worn).
- Determining other descriptors of people (such as hair colour or sex) passing in front of the camera.
- Counting and determining other objects passing through the scene.

1.2.2) Motivation:

Human motion analysis is receiving increasing attention from computer vision researchers today. The interest is motivated by a wide spectrum of applications such as surveillance, man-machine interfaces, content-based image storage and retrieval and video conferencing. The capability to monitor human activities in security-sensitive areas such as airports, borders, building lobbies and car parks (to mention a few) is of great interest to the police and military, not to mention the commercial security potential. With the development of digital libraries, the ability to automatically interpret video sequences will save tremendous human effort in sorting and retrieving images or video sequences using content-based queries [**Aggarwal & Cai**]. Some examples of the direct applications of this project are:

1. Advertising: e.g. where the numbers of people who walk past billboards would be of interest to marketing companies.
2. Statistical studies: e.g. where the number of people who use the train at a particular time would be of interest.
3. Surveillance video processing: The sub-goals of this project allow for the effective processing of surveillance video, an extremely monotonous task for humans (and hence frequently unreliable), to discover if certain types of people (based on a provided description) are to be found in the video sequence and if so at what times.
4. Content-based queries: again one of the sub-goals of this project is to be able to analyse a video sequence and search for people matching a provided description which would be of great value to the area of content-based queries on video sequences.

Chapter 2 – Background Information

2.1) What we have:

2.1.1) General Background Information:

There are currently a number of video processing packages in the market, which are designed with various different goals in mind. There are real-time licence plate recognition systems [**citysync**], intruder detection systems and surveillance video enhancing systems that do some of the things that this project aims to achieve. There are also multiple systems associated with the detection of human faces. Most of these systems are associated with the recognition and identification of a particular human face at a short range. This is not feasible in this project as the camera will be monitoring a changing scene with many people walking towards and away from the camera at various distances. There are some projects which try to recognise faces with these (or similar) constraints such the Intelligent Computer Interface (ICI) system which is a long term research project aimed at combining computer vision (face recognition), animation, speech recognition and speech synthesis in an AI framework [**Hjelmås, Lerøy, Johansen**].

Dr. Janne Heikkilä at the University of Oulu in Finland is developing person-counting software based on a "snake" technique for finding boundaries in regions of frames of a video sequence and subsequently identifying contours that could correspond to a human shape. [**Janne**] The snake technique is a contour modelling technique where the "snake" refers to a deformable contour that moves under a variety of image constraints (which tend to be local) and object-model constraints. The representation of a snake is v(s) = (x(s),y(s)) where s runs from 0 to 1 over the perimeter of the snake. The snake is controlled by minimizing a function that converts high-level contour information like curvature and discontinuities and low-level image information like edge gradients and terminations into energies. It is basically a

complex edge finding algorithm, which traces edges of objects and finds humans by their unique shape (a spherical head on a body). *For example:*

The image above is one frame from a video sequence that shows how the algorithm has identified two moving objects; it should be clear how the peaks in the red outlines represent the two humans.

There is also a feasibility study being done by the University of Minnesota and Honeywell to determine the effectiveness of combining infrared and visible spectrum cameras to automatically count the number of passengers in vehicles [**Minnesota & Honeywell**]. The reason for this study is to try and come up with a computer vision algorithm that will be able to reliably determine the number of human passengers in vehicles that pass the camera. The current method involves people sitting near the road and observing traffic to collect the data. This causes a number of problems:

- The people collecting the data sometimes cause congestion when passing motorists slow down to "gawk"
- It is difficult for people to conduct accurate data collection for extended periods of time or during bad weather
- It is costly to manually collect data, which limits the number of data samples that can be taken

In terms of actual work in the field there have been a number of papers and research projects on two of the main areas of interest (the tracking of moving objects and human recognition) as well as papers on the particular difficulties that occur with this sort of project. For example, there has been some study of the problem of occlusion (where one object obscures the cameras vision from another object) that is of concern in this project in respect to the correct counting of people in groups [**Khan & Shah].**

This paper details how people are represented as a mixture of gaussians in spatial and colour space. Each person is modelled as a set of classes, where each class has a spatial component (x,y) and a colour component (Y,U,V). A class is thus represented by a 5-dimensional gaussian distribution. The classes are tracked from one frame to another using a maximum *a posteriori* probability approach. This approach, according to this paper, lets the system deal "implicitly with occlusion, and is able to correctly label people *during* occlusion".

There are also object-tracking algorithms that concentrate more on colour space. For instance there is a paper [**Feiguth & Terzopoulos**] that details the development of a simple and very fast method of object tracking based exclusively on colour information in video sequences. The research is based on tracking regions of similar normalized colour from frame to frame and deals with the occlusion problem using an explicit hypothesis-tree model of the occlusion process, which the authors believe allows them to achieve their goal of "develop[ing] a tracking algorithm capable of tracking multiple objects in real-time at full frame rate".

Specific Background on relevant Techniques:

2.1.2) Optical Flow Analysis:

Optical flow reflects the image changes due to motion during a time interval *dt*, and the optical flow field is the velocity field that represents the three-dimensional motion of object points across a two-dimensional image [**Kearney and Thompson**].

If a series of images of a scene are taken in time, and there are moving objects in this scene, then analysing and understanding the difference between these sequential images can generate useful information about the motion of these moving objects within the scene.

For example, given an image of a moving car, deciding which pixels in the image represent motion can help to decide which pixels belong to the car, and which to the static background.

Studying the motion in detail, the following questions can be answered by optical flow analysis:

- How many moving objects there are?
- Which directions they are moving in?
- Are the objects undergoing linear or rotational motion?
- How fast they are moving?

Optical flow fields have a natural application in tracking objects (if the answers to the above questions can be obtained then there is a nice basis upon which to track objects correctly). As such there have been a number of developments in object tracking using optical flow, [**Adiv**] has analysed the resultant optical flow fields generated from the basic optical flow calculations to calculate 3D motion estimates for several moving objects. There is some difficulty, however, in analysing the results from optical flow fields to determine the movements of interacting 3D objects (such as

groups of people for example) and the fundamental object segmentation decisions need some further image information.

2.1.3) The Kalman Filter:

The Kalman filter is an algorithm used to perform filtering on a process model with linear dynamics and linear observations, both subject to Gaussian noise. A development of the Weiner filter made in the 50's, it basically provides a mathematical theory for the recursive estimation and prediction of an unknown time function (with error estimation with each prediction) based on another, observed one. In vision applications which track objects, the Kalman filter can be used to model the behaviour of each moving pixel, with predictions of its state and position in the next frame, given information about the pixel's initial state, velocity and acceleration.

The Kalman filter works by estimating an unknown vector y, based on an observed data vector z, by finding the mean-squared estimate, where the estimate $\hat{y}$ is chosen so as to minimize the expected value of the Euclidean norm squared of the error (the best fit). The Kalman filter implements a recursive least squares fit to the data, given some assumptions about the data [**Nickels**]. A detailed description of the operation of the filter is not relevant here but can be found elsewhere [**Nickels**], [**Kohler**].

The use of this prediction of object position is of certain value to a tracking application. Once objects are identified (by background subtraction for instance) Kalman filter prediction would be able to tell where these objects are likely to appear in the next frame. This means that only certain parts of the next frame need be searched for the object, which would reduce the number of pixel operations needed (and hence increase efficiency). Also any noise in the image generated from background subtraction, outside the predicted area, would be ignored and so have no effect on the tracking system.

The Kalman filter has been used in a number of visual tracking applications to date (since the early eighties). The first tracking use of the filter was in 1980 where it was

used to track a number of sonar targets underwater [**Hallam**]. In visual tracking, a least squares filter derived directly from the Kalman filter was used to track known 3D objects in 1982 [**Gennery**]. Since then it has been widely used, especially the Extended Kalman Filter (EKF) in many different computer vision projects. Some of the more recent and interesting uses of the Kalman filter (with particular relevance to this project) have been:

- The tracking of an active contour [**Blake, Curwen & Zisserman**],
- The tracking of the bounding rectangle of a vehicle along an assumed ground plane [**Matteuci**],
- An active contour model with a Kalman filter used to track the 2D silhouette of an object in an image sequence [**Stark & Fuchs**].

In the specific area of tracking people, there is a paper [**Kohler**] that uses the tracking of human interactive motion as an example of how to initialise and model a Kalman filter for translational motion, but the system is subject to a number of constraints (very complex, non-general Kalman matrix initialisation and vision-based sensoring must be estimated carefully).

Although considered for this project, it was felt that the time needed to get a fully working Kalman filter to track people and deal with occlusion, could not be justified as tracking was not the sole goal of this project. A less sophisticated (but perhaps more robust) tracking method would be needed that could be tailored specifically to *people* moving in an image sequence.

2.1.4) Neural Networks:

There are a number of papers that detail face detection using neural networks. Neural Networks (Artificial Neural Systems) are information processing devices whose design is inspired by studies of the brain and nervous system structure and function. Much of the Neural Network terminology reflects this relationship to biological nervous system structure.

As an information processing technology, Neural Networks accept several inputs, perform a series of operations on them, and produce one or more outputs. The network connections carry numeric, as opposed to symbolic, data. The units operate only on their local data and on the inputs they receive via the connections. Most Neural Networks have a "training" rule whereby the weights of connections are adjusted on the basis of presented patterns. In other words, Neural Networks "learn" from examples, just as humans learn to recognize a cow or an insect from experiences with cows and insects that allow generalization to others of their kind **[Lemon, Rhodus & Hall]**.

Neural networks are easily applied to pattern recognition and classification problems and so are of particular interest to this project. They have been fairly extensively used in the area of face recognition, there is a paper which details face recognition based on a hybrid neural network, which combines local image sampling, a self-organising map neural network, and a convolutional neural network [**Lawrence, Giles, Tsoi & Back**]. A neural network based solution for face recognition has been developed that works effectively for still greyscale images [**Rowley, Baluja & Kanade**], and there are systems that use colour information with neural networks [**Porter**] but the colour information is not used effectively - it is provided as part of the networks input and not used as a separate feature. Colour as a separate feature used in conjunction with neural network classification would add a level of abstraction that would increase accurate classification.

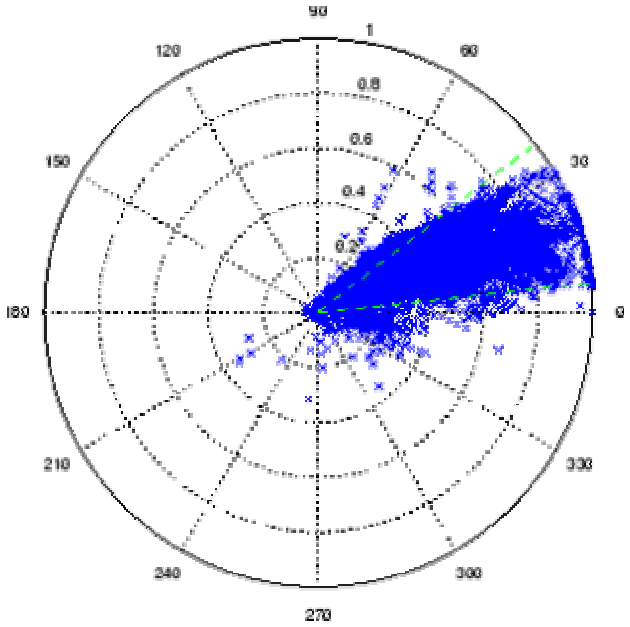
Some of the more relevant papers are those that deal with face detection (multiple and single) at a distance, as opposed to specific face recognition. For example, a paper originating from Carnegie Mellon University in Pittsburgh [**Rowley, Baluja & Kanade**] presents a neural network based face detection system that examines small windows of an image and decides whether that window contains a face or not. This system works only with grey-scale images but "is able to detect 90.5% of the faces over a test set of 130 images, with an acceptable number of false positives". The drawbacks to this system are that:

- It does not make any use of colour (which for human detection would seem to be able to provide useful information).
- It doesn't appear to handle oddities such as glasses or sunglasses.
- The network is highly calibrated to frontal views of still images (it is not as robust at detecting side-on views of peoples faces).

Example of Neural Network face classifier at work (some of the false positives can be seen and also the problem with side-on faces). If colour were also used, better results may be obtained – skin colour has quite distinctive hue characteristics:

2.1.5) Skin Colour analysis (in HSV Space):

One of the main issues in this project will be the detection of humans, if any, in a particular image. One of the ways to do this is by finding pixels that could represent human skin. The detection of skin colour in images is a very useful technique in computer vision for detecting and tracking humans. "As a visual cue, skin colour is robust and inexpensive to compute, making it useful as an attention-focusing mechanism for more expensive computations such as humanoid shape determination." [**Raja, McKenna and Gong**]. Skin colour from all ethnic groups cluster tightly in hue-saturation (HS)-space.



Skin pixels plotted in HS-space. (Hue is the angle $\boldsymbol{\theta}$ *with red at 0°; Saturation is the angle* $\boldsymbol{\alpha}$*. The tight cluster can be seen between* $\boldsymbol{\theta}$ *= 6° and* $\boldsymbol{\theta}$ *= 38°)*

When modelling skin colour, the problem is basically a binary classification problem. Each pixel is classified as skin or non-skin based on its colour components. Classifying skin pixels into Skin/Not Skin will not be perfectly accurate all of the time as there will always be elements of various background features that could fall into the skin "colour-space". Also, a face will sometimes contain unusual colour features depending on illumination that will be indistinguishable from certain white-ish regions in the background. Therefore skin pixel determination alone will not be

sufficient to identify people but will act as a good starting-point for the detection of humans in an image sequence. It can also act as a good localizing function for the initialization of another detection algorithm (for example a classifier or template matcher that will look for faces in an image will only have to scan areas of high skin pixel concentration).

Although an algorithm for detecting skin pixels based on the above description was written and implemented, it failed to be a reliable cue for humans for a number of reasons. The two main reasons were that if someone was walking away from the camera there may be no skin coloured pixels at all on the object and secondly given the size of video file used (320 X 240) for the sake of computational expense, there were frequently no skin pixels detectable on a person walking toward the camera until that person was quite close to the camera. For these reasons, the skin pixel identification algorithm was not integrated into the system.

2.2) What we need

Data:

In order to be able to test the algorithms that were written at each stage, the first thing that was required was some video footage of various objects, and more specifically people moving through various background scenes. The first video sequences were taken from the web, and subsequently, when more specific scenes were necessary, a digital camera was used to film at various places within Trinity and video sequences of people walking to and from the camera in various numbers were taken.

Chapter 3 – Plan & Design

3.1) Introduction

In this section the planned development of the project will be discussed. The software lifecycle for the project will be briefly introduced and reasons for its use made clear. The overall algorithm is then broken down into the stages in which it was tackled and then each stage is discussed in detail. There will be no technical definition of the algorithm used at each stage, as this will follow in the implementation section, simply a discussion of what is meant by each stage, the basics of its implementation and some results where applicable.

3.2) Software development lifecycle

There wasn't really a need for a comprehensive software lifecycle for this project (as most development lifecycles deal exclusively with large-scale applications with a client-oriented approach and team-based development). Nevertheless a certain amount of planned design was considered necessary. As such, a plan based on an amalgamation of the Coverdale Systematic approach (discussed earlier) with staged development (similar to the "Developmental Stages" concept of the Waterfall Software Lifecycle **[Treglown]**) was used. Each stage was planned as a distinct algorithm, which could be written and tested separately, and then incorporated into the main project. The rest of this chapter will deal with those stages individually.

3.3) Stages of development

The development of the counting people algorithm was taken in discrete stages (as per the lifecycle discussed above). The problem can be broken down in quite a reasonable fashion, delineating various necessary tasks for separate development and

testing and then putting them all together to form the main application. As such, the problem was divided into the following stages:

1) Find moving pixels (pixels which do not match the background image) and display as a binary (moving or not moving) image
2) Remove unnecessary noise from this image
3) Connect up moving pixel regions which are very close to one another
4) Analyse moving pixel information to find homogenous regions for further processing
5) Track objects that are of a sufficient size from frame to frame
6) Create object groups when two tracked objects collide (one object may occlude another)
7) Generate object histograms for each object when on its own in order to reason about objects after occlusion
8) After two objects, which have come together, separate again, use the histogram information of each object to determine which is which

These eight sections cover the main tracking and counting algorithm. Some enhancements that were added in order to improve the counting accuracy are:

1) An adaptive background model to solve the problem of hard-coding thresholding parameters (section **3.4.9**)
2) An estimation function to estimate how many people are in an object based purely on the objects geometric characteristics – this is necessary because when a moving object is first seen, it may be two or more people walking close enough to each other to be one connected region in the moving pixel binary image. (section **3.4.10**)
3) Shadow removal technique to stop shadows distorting moving objects or being misclassified as moving people (section **3.4.11**)

A proposed technique for determining which object is which in a group of people during occlusion will also be discussed briefly (section **3.4.12**). Although all the necessary software techniques are in place for this, it was not thought necessary to implement because the added information it provides does not have much meaning for the goals of this project (if two people come together into a group, the knowledge that there are two people in the group is all that is needed for the purpose of counting, not

which is which). It is however an integral part of most tracking systems and so is worth mentioning.

And finally, a separate database application was developed to show one of the proposed applications of the project – allowing a user to search through a video sequence for people matching a particular description. Some database functionality was added to the c++ tracking algorithm and the functionality for determining what "colour" clothes people were wearing (a non-trivial method from a computers perspective) was added to the project.

3.4) – Overall Algorithm Definition

3.4.1 – Background subtraction:

Background subtraction is a method typically used to detect unusual motion in the scene by comparing each new frame to a model of the scene background. Given a background image of a scene, moving pixels can be located simply be obtaining the absolute values of the difference between the corresponding pixels in current frame and the background (or current frame and the last frame), and large values in the difference map then indicate locations of change. The difference map is usually binarised by thresholding it at some pre-determined value to obtain a change/no-change classification (see example overleaf). The threshold value is critical, since too low a value will swamp the difference map with spurious changes (noise), while too high a value will suppress significant changes. The proper value of the threshold is dependent on the scene, randomly fluctuating camera levels, as well as viewing conditions (e.g. illumination) that may change over time.

Fig 4(a) – Example of background subtraction:

The current frame of a video with two people visible

The background image for this scene

The binarised moving pixels image resulting from the background subtraction operation

A model for background subtraction based on a nonparametric technique for modelling the background of a scene is presented by [**AElgammel, DHarwood, LDavis]**. The approach is based on kernel density estimation of the probability density function of the intensity of each pixel given a sample for this pixel. This was considered an overly complex solution to background subtraction and although the thresholding function was later made more complicated by using an adaptive background model (see 3**.4.9**), the basic background subtraction algorithm was found to be sufficient for this application.

3.4.2 – Noise Reduction (Convolution)

After background subtraction has occurred, there will inevitably be pixels in the resulting image that show up as having moved, but are not a valid moving object. This can be due to tiny movements of the camera during filming, the camera's own internal random Gaussian noise distribution in its R,G,B channels during the saving of the digital image or some artefacts of the compression process used to create the video file for example. These pixels do not represent valid moving objects, and yet will show up as "moving pixels". This is referred to as noise. Although the number of these noisy pixels can be significantly reduced by accurate thresholding, there will always be some noise generated (and choosing a threshold that will give all the correct moving object pixels and very little noise is extremely difficult).

As such, a noise reduction algorithm was applied to the image generated by the background subtraction. The aim of this algorithm is to eliminate the random, unwanted pixels that are not part of a large moving object, and yet keep almost all of the pixels that are part of a moving object. To do this, a convolution mask is used, which is a rectangular shape that defines a neighbourhood around a target pixel on which to reason about this pixels validity. The mask used will be:

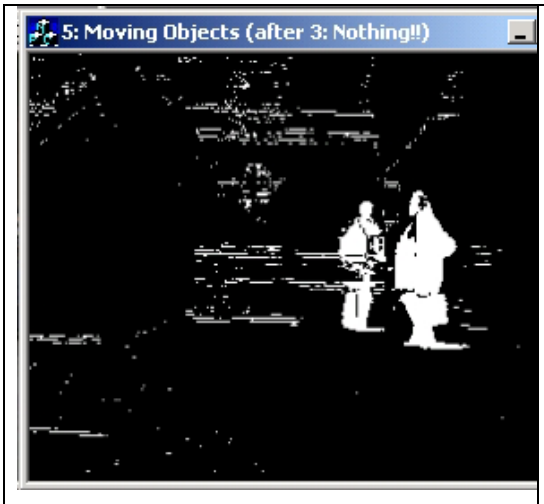
H=1/16 *

1	2	1
2	4	2
1	2	1

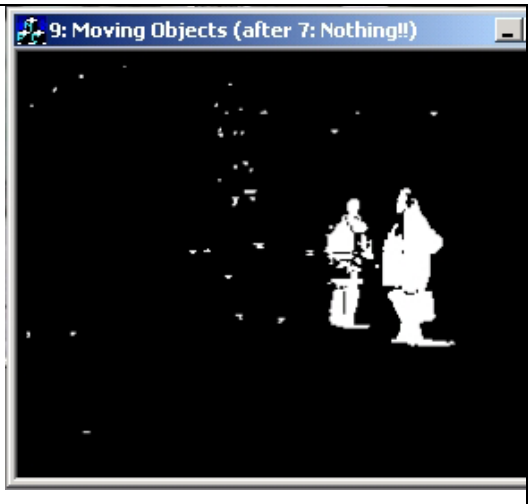
Convolution mask – each point in the mask is multiplied by the underlying image point.

The pixel being analysed will be the centre pixel, which will get the most weight. Since the image being analysed is a binary image each of the surrounding pixels will either add the weight in the mask to the total if it is a moving pixel or add zero if it is not. If the resulting total is closer to 1 the pixel is said to be part of a moving object, if not the pixel is merely noise. In simple terms, unless the moving pixel is connected to other moving pixels, it is considered noise. Only pixels that are different from the background in the first place (thresholding) are checked.

Example of noise reduction:



All Moving Pixels (there is a lot of unnecessary noise here which does not correspond to a valid moving object)



This is the same frame with the noise reduction algorithm turned on. The noise has been dramatically reduced without affecting the moving objects themselves.

3.4.3 – Binary Morphology (Opening & Closing)

Binary morphology is a subset of basic mathematical morphology. It is based on "the algebra of non-linear operators operating on object shape" [**Sonka**]. Morphology is used in many different areas of computer vision and image analysis such as noise filtering, shape simplification, skeletonising, thinning, thickening, convex hull calculations, object marking, object segmentation, and the quantitative description of objects [**Sonka**]. The principle behind mathematical morphology in computer vision is to treat a 2D binary image as a point set based on an arbitrarily chosen origin (in this case the top left corner of the image), which is given the point value (0,0). To apply a morphological operation a particular structuring element (similar to a convolution mask) is moved systematically across the image. The application of this element basically means computing the relation between the input image (point set X) with the structuring element (point set Y) with respect to a local origin, which is the current point in the image for which the output is being computed. In essence the "next state" of each point x in the image is based on the current states of the points in the input image that fall within the structuring element as applied to the local origin.

The binary morphological operations that will be used in this project are opening and closing with an isotropic structuring element. The result of opening an image is to remove small noisy image elements and the result of closing on a binary image is to connect objects that are close together, fill up small holes that may appear in objects and smooth object outlines. These operations will be used after the background subtraction and noise removal stages to:

- Remove objects too small to be valid trackable regions
- Connect moving objects that have been split by a noisy background
- Fill in the holes within moving objects that may appear due to the background subtraction technique or noise.

The opening and closing operations used will be called from Intel's Image processing library and so will not need to be discussed in the implementation section.

Example:

This is the same frame as above (after noise reduction) with morphological operations applied. This further reduces the noise and connects disjoint parts of the same object (fills in gaps).

3.4.4 – Connected Component Analysis

Connected component analysis is a method used to find homogenous regions in an image – in this case a binary image. Black pixels are considered to be background pixels and so any white pixels indicate foreground pixels (the objects the program is interested in). The essential idea behind connected component analysis is to connect all the pixels that touch other white (foreground pixels) together in order to form a region. So instead of foreground pixels, there are now foreground regions.

The fundamental method is relatively simple, although there are many ways of implementing it. An iterative approach was used in this project. The image, in this case the background subtraction image of the current frame with noise reduction completed, is searched through pixel by pixel, from top left corner to bottom right and any white pixel is labelled (i.e. given a unique number). The label is either a new label if none of the surrounding pixels (that have already been passed) are labelled or else one of the surrounding pixel's labels. If there is more than one distinct label around the pixel, an "equivalence" is noted between them. After all the pixels have been thus labelled, all the equivalent labels are relabelled so that connected pixels

have the same label and so become the same connected component or region (see example below).

Connect Component analysis example [Sonka]:

(a) *Original Binary image*

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	1	0
0	1	1	0	0	0	1	0	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) *After initial labelling*

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	3	3	0	4	0
0	5	5	5	2	2	2	0	0	3	0	0	4	0
0	0	0	0	5	0	2	0	0	0	0	0	4	0
0	6	6	5	5	5	2	2	2	2	2	4	4	0
0	0	0	0	5	5	5	2	2	2	2	2	4	0
0	7	7	0	0	0	5	0	2	0	0	2	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

(c) *Equivalent labels matched & relabelled correctly*

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	1	1	0	2	0
0	2	2	2	2	2	2	0	0	1	0	0	2	0
0	0	0	0	2	0	2	0	0	0	0	0	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	0
0	0	0	0	2	2	2	2	2	2	2	2	2	0
0	3	3	0	0	0	2	0	2	0	0	2	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

The regions identified by connected component analysis form the basis for object identification and object tracking.

3.4.5 – Basic Tracking

The basic tracking used in this project is loosely based on a robust algorithm described by [**McKenna, Jabri, Duric & Wechsler]** in their paper on tracking interacting people. In that paper the authors treated multiple regions, tracked individually (which satisfy the rules of being close to one another and having projections that overlap in the x-axis) as a person. In order to simplify things slightly, each region identified by the connected components analysis (see section **3.4.4**), that is of sufficient size, is treated as a possible person – this is a reasonable change to the papers algorithm as morphological operations are used on the moving pixels image which serve the same purpose as the rules described above to some extent.

Subject to this region satisfying the aspect ratio tests for a human shape (discussed further in the estimation function **3.4.10**) the region is deemed eligible for tracking. A region, which is labelled trackable as a person, must remain trackable for at least three frames before it is actively tracked. When a region is being actively tracked, all the current characteristics of the region, such as height, width, area and most importantly the region's centre point, are stored in a Region Monitor object in the program's memory (for a definition of this object please see the class descriptions in the implementation section **4.2**). When a current frame is being processed, and a region is identified as trackable, all the currently held regions are checked and the one in memory whose centre is closest to the centre point of the current object (must be within a certain minimum distance) is assumed to be the same object one frame later. It must be noted that it is assumed that the difference in centre positions will be very small, as each frame will only differ slightly from the last one. The object held in memory is updated with its new characteristics and the program moves on to the next region.

This is a relatively effective method of tracking as no two objects of sufficient size to represent a human can have the same or very similar centre points a single frame later without touching each other and so becoming a new much larger region representing more than one person (this will be handled by "grouping" regions discussed in the next section **3.4.6**). It also has the advantage of being computationally inexpensive

and scalable to far more complicated reasoning (such as aspect ratio, skin colour analysis, prediction algorithms) which can all be "bolted on" to enhance the tracking if necessary. In practice there was never any need for anything more complicated – no video footage tested of people moving singly (two or more people walking together without overlapping) ever caused the program to miscalculate which object was which during frame-to-frame tracking. It could be argued that in certain non-ideal circumstances (such as one object accidentally splitting into two regions) that this method is not fault-tolerant enough to cope. That situation, however, arises out of imperfect background subtraction so isn't really a fault with the tracking algorithm per say and should be dealt with at a lower level.

3.4.6 – Grouping People

A group consists of one or more people walking together and whose object regions overlap so that they become one connected component (see example below). A group may consist of two known distinct objects, which have come together after being tracked singly already, or a group may enter the scene as a group and the program will know nothing of the internal objects in the group. The tracking algorithm treats both these cases separately.

Example of region grouping

<table>
<tr>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>The current frame (notice how the two people "overlap" at their feet)</td>
<td>The resulting regions image. This is one region corresponding to two distinct people and so represents a group</td>
</tr>
</table>

In order to tackle the former situation, a new class was defined to deal with groups of objects and a tracked region representing one person now became a group of size one (for more information on the Object Group class please see implementation section class definitions **4.2**). Since each individual person is now a group of size one, when two people collide (regions merge) the second person can be added into the first persons group to form a group of size two. Each group has its own internal region as well as the regions corresponding to each of the people it is holding. This means that a group of any size can be tracked as an object in its own right (the internal objects aren't tracked individually until they break away from the group again).

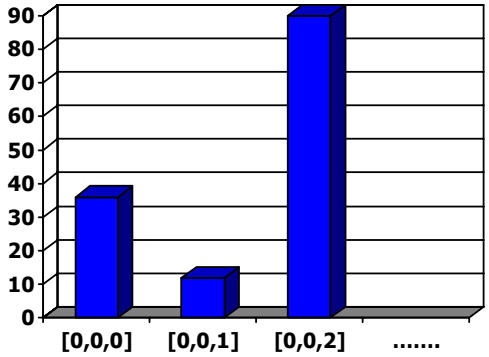
In the second situation, a group of size one is created but an estimation function is used (discussed further in section **3.4.10**) to let the program know that the object being tracked is probably more than one person. No reasoning is done on the internal objects of the group while they remain as part of the group (since the main goal of the project is to count people and this is accomplished effectively without any information needed about a group's characteristics). If any object splits from the group, its size is estimated (as it may still remain a group of more than one person) and it initialises a new region monitor and is tracked separately as before.

3.4.7 – Generating Colour Histograms

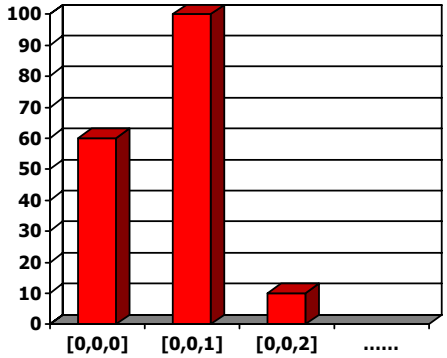
Once a method of grouping tracked objects when they merge was developed, a method of determining which object is which when the region splits again was required. In order to accomplish this there was a need to model each person's appearance, so that judgements can be made about which region in the current frame belongs to which object in the group that has just split up. A method based on colour histograms was decided upon, as they are computationally inexpensive, can be updated adaptively if necessary and have been successfully used before in object tracking [**Martin, Devin, Crowley**]. They fit in nicely with the basic tracking algorithm and are used, although quite differently, for occlusion resolution during tracking [**McKenna, Jabri, Duric & Wechsler**]. Since histograms are used for reasoning post-occlusion, a method of histogram intersection developed for colour indexing [**Swain**] was used to match two object histograms (discussed in the next section **3.4.8**).

A colour histogram is basically a structure that holds the number of occurrences of each possible combination of RGB values (so H(x) simply counts the number of occurrences of x = (r,g,b) in the image). So for an 8 bit image (256 possible values for each channel) there would be 256^3 histogram bins. Since this is a very large number of bins to have (and as this level of detail is far more than needed), the quantisation of the object being histogrammed is reduced to 4 bits (16 values per channel) and so only 16^3 or 4096 histogram bins are available. Colour Histograms can be conceptualised as Bar Charts and their comparison is analogous to comparing how accurately two bar charts distributions match each other (see example overleaf).

Example – histograms as bar charts:



Histogram holding the number of Occurrences of x=[R,G,B] in an image



These two sample bar charts differ significantly from each other, which is essentially the basis for the histogram comparison algorithm

The histogram algorithm will then simply take each object being actively tracked, compute its histogram and store it in the objects region monitor. This means that the program now has an appearance model based on colour associated with each object being tracked that it can use to re-identify that object after it has come out of a group.

Note, only object groups that have one region are histogrammed as it would not be possible to reliably identify a group of people after a larger group has split up because multiple groups of multiple people merging may give rise to a different combination of people in a group after the resulting split.

3.4.8 – Using histograms to aid tracking

The actual method used to determine which object is which after a region grouping has occurred is a technique known as histogram intersection. This is quite a simple algorithm that matches the counts of corresponding histogram bins in two different histograms. The intersection value is then normalised in the range [0,1] by dividing by the number of pixels in the model histogram. This means that the resulting value can be considered as a probability that the one histogram matches another. As each of the region monitor objects being tracked have a histogram associated with them, when an object splits off from a group the region corresponding to that object is histogrammed and the result compared to the histogram in each object in the original group via histogram intersection. This results in a probability value for each region in the group (i.e. how likely is it that this new region matches each region in the group). The region in the group with the highest probability is chosen as the "splitter" and is removed from the group and added to the tracking algorithm as a group on its own.

Histogram intersection is a robust solution that is very fast (requiring time proportional to the number of histogram bins) and foregoes the need for more complicated matching methods. The method has applications in occlusion detection as well if necessary as the non-occluded areas of colour should still be of sufficient size to allow for object recognition (although this wasn't implemented for this project).

The only possible problem that may theoretically occur using this method would be if there were two people who were dressed identically (in some kind of uniform for example) but even slight difference should increase the probability enough for recognition so this was considered an acceptable constraint.

3.4.9 – The Adaptive background model

One of the issues that frequently crop up in the computer vision area is that of "threshold values". In this case a threshold value is used for the background subtraction method described in section **3.4.1** (recall the test to see if the difference between a current pixels RGB values and a background pixels RGB values was above a certain threshold the pixel was labelled as moving). Although this works very well, there is an inherent problem with the generalisation of this method. How can a single threshold value work in many different circumstances and on many different background images? The answer is that it can't and in the end any method based on a hard-coded threshold value will be optimal for a *particular* video file and background image (but will probably work *acceptably* well for most other cases).

Another foible of the background subtraction method is the difficulty it has in handling complex backgrounds where some of the backgrounds colours match some of the colours in the people moving in the foreground. This can cause the mislabelling of region pixels that are moving as background pixels because they don't differ sufficiently from the underlying background to be considered moving by the algorithm (see figure 5**.2.1** in section **5.2** for an example of the problem).

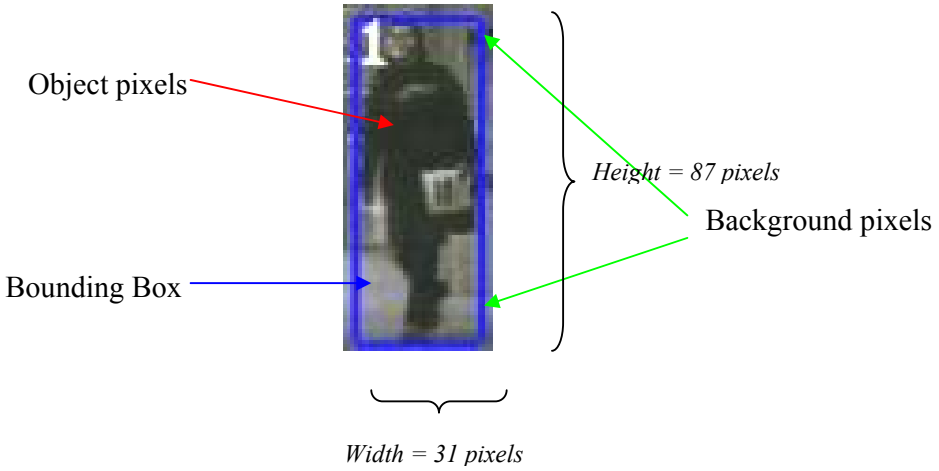
In order to address these two issues (the latter only to a certain extent) an adaptive background model was created. What an adaptive background model does is store the mean value of each background pixel, updating frame by frame. It also stores the standard deviation from this mean for each pixel. Any subsequent test will use this information instead of a pre-defined hard-coded threshold. So instead of checking to see if the difference between a current frame pixel and a background pixel is above a threshold, the program checks to see if the difference between a current frame pixel and the mean value for that background pixel is more than 3 times the standard deviation for that pixel. This eliminates the need for a pre-defined threshold and allows the algorithm to dynamically determine the threshold to be used for each pixel. This method also helps with the second issue in background subtraction because the adaptive model will tend toward a more accurate threshold for each pixel instead of a

globally (necessarily larger than necessary for a significant number of pixels) chosen threshold value.

3.4.10 – The Estimation of group size

The estimation of group size is needed to avoid tracking an object, which represents a group of people that has entered the scene as a single connected region, as one person, thereby miscounting the number of people currently in the scene. The function takes each region being tracked and analyses certain characteristics such as aspect ratio and object pixel ratio and makes an estimate of the number of people within the group. Absolute characteristics such as region area cannot be used because the total area of a two-person object can be the same as a one-person object if each is seen at different distances from the camera. The ratio of the height of a region, representing a single moving person, to its width is around 3 (see example below) and the ratio of a region representing 2 people (walking roughly side by side) is about 1.5. Three people together is about 0.8. The object pixel ratio refers to how much of the pixels within the bounding box of the current region are in fact object pixels and how many are background. A human shape is not geometrically rectangular so it would be expected that any bounding box around a region representing a human would have a certain number of object pixels in it (see example overleaf).

Example of region bounding box around a humanoid shape:



The estimation function estimates up to a maximum of three people in a group as after this the number of people estimated will become increasingly inaccurate as large groups rarely all walk side-by-side. Besides, it would be unusual for a group larger than three people to enter a scene as one connected region without at least one frame where the regions diverge and different sub-groups can be identified.

3.4.11 – The problem of Shadows

Shadows are, unfortunately, an inherent part of any moving object. They are generally, but not always, attached or connected to that object. There are many degrees of shadow, from light shade changes to very dark, well-defined patches. The major problem that shadows pose to this project is that they can appear as part of the objects that are being tracked. Since they are very close to, if not directly connected to, a moving person, they can end up forming part of the same object region (especially after the morphological operations explained above in **3.4.3**), which can lead to a significant error in the size and aspect ratio of an object that makes the accuracy of group size estimation difficult if not impossible. If the shadows are not quite close enough to be connected to a moving person, and are of sufficient size, then they can be quite easily mistaken for a second person. (Since shadows will roughly be of a similar shape and size to a person being tracked, it is easy to see how difficult it would be to differentiate between moving pixels associated with a person and those belonging to their shadow from the perspective of a binary moving pixels image).

As can be seen from the following example (overleaf), the shadow created on the wall by the man in the blue top on the right hand side of the first image looks in the moving pixels image very similar to a third moving person. After noise reduction and morphological operations, this shadow will end up as a clean well-defined human shaped region and will be indistinguishable from a third person.

Fig 1: The problems shadows cause:

The current frame of video (notice the shadow projected onto the brick wall by the person in blue walking on the right of the two people)

This is the moving pixels image generated using the basic moving pixels algorithm. It is evident that there is a significant problem here for the recognition algorithm

Clearly, it is necessary to remove shadows before the region generation takes place, that way by the time the algorithm is doing its higher level reasoning on the objects it will not have any third object or region to worry about. It will also mean that parts of a connected moving region that correspond to shadows will not be present in the binary moving pixels image and so will not distort the regions correctly corresponding to a human. This is not an easy task. The program must be able to understand that certain types of moving pixel are to be ignored during moving pixel generation. In order to do this it is necessary to find something about the "shadow pixels" that differentiates them from other moving pixels.

There are a number of proposed ways of doing this ranging from the extremely complex to the deceptively simple. The simplest way, apparently, to remove shadows is to look for a significant change in intensity with no corresponding change in hue or saturation (in the HSV colour space). This simply doesn't work very well. When implemented it only slightly reduced the shadow gain (on certain background types) and it also caused a not insignificant loss in actual moving object detail. A second method [**McKenna, Jabri, Duric & Wechsler]** that uses "chromaticity" (which is in effect a version of the RGB colour space with intensity information removed) was also relatively ineffective and prone to necessary "parameter tweaking" in order to work at all. An adaptive version of the chromaticity model used to remove the hard-coded parameters also didn't work properly for many combinations of shadow types.

None of the above methods reliably dealt with either complicated background scenes or hard, well-defined shadows on simple backgrounds. In the end, shadows became the crux of many of the difficulties and misclassifications that the program was making and so a new solution to the shadow problem seemed increasingly necessary. The resolution of the problem lay in a complex set of rules based on a number of parameters of both the background and the current frame in HSV colour space. This new algorithm makes use of the magnitude of the difference between the saturation channel and the intensity channel of each pixel in the image. Depending on the type of background pixel (whether the difference between its S and V channels is very large) one or the other of the two methods is used:

Method A: If the difference is small, then the rules based on comparing the difference between the backgrounds S and V channels and the difference between the current frames S and V channels is used (much in the same way that the original moving pixel algorithm works). If there is a large change in the hue channel or a large change in the *difference* between the SV channels then the pixel is said to be a moving pixel. This eliminates shadow pixels because a shadow cast on a simple background tends to change the intensity and saturation channels in the same direction and by roughly the same amount whilst leaving the hue relatively untouched.

Method B: If, however, the background pixel has a large difference between its own S and V channels then a somewhat more esoteric solution is necessary. Shadows falling on these surfaces tend to have a more unpredictable effect on hue. The hue might change quite dramatically depending on the strength of the shadow. Also if the saturation channel is already very low and the intensity quite high, the shadow will cause a decrease in intensity with little change to the saturation. This means that the above solution (though reducing shadows to some extent) will not be able to remove enough of the shadow to avoid object distortion. A different solution is used in these cases. The current frames V channel is compared to the backgrounds S channel (VSDiff) and the current frames S channel is compared to the backgrounds V channel (SVDiff). If both these values are large then this pixel is deemed a background pixel (the value of "large" depends on the difference between the backgrounds own V and S

channels so is not hard coded). Otherwise, subject to this pixel also satisfying the original moving pixel criteria (detailed in **3.4.1**), it is deemed to be a moving pixel. This method takes advantage of the fact that since this pixel in the background has a high V-S difference in its own V and S channels, if the current pixel is not a moving pixel the differences between it's V values and the backgrounds S will also be high (and the same for the S – V difference). If, however, it is a moving pixel, the V or S channel will most likely have changed sufficiently to fail one of the V-S tests with the background. Shadows are removed because they do not change the V and S channels sufficiently in the direction of each other and any change they may have made to the hue channel is now being ignored.

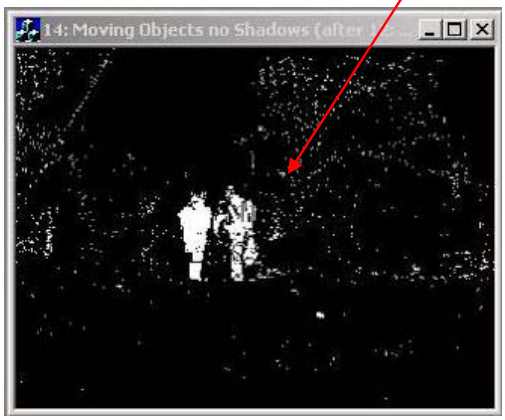
Since neither of these methods works in the domain of the other (the second method is useless when the background is very plain and vice versa), combine the two sets of rules are combined together and a check is made to see how different the background's S and V channels are before choosing a set of rules to determine whether a pixel is moving or not. The resulting solution reliably eliminates shadows of all degrees found in the test videos with little or no degradation of actual moving object definition. There are few stipulations in using this technique, a modest assumption that people moving in a scene with a non-simple background do not wear colours whose saturation match the backgrounds intensity, and whose intensity match the backgrounds saturation. This seems a workable assumption, since in the original moving pixel identification it must be assumed that the moving people's clothes don't match the backgrounds colours anyway.

A more detailed explanation and example of how this works will follow in the implementation section and a detailed example of the solution in action is given in Appendix A "Shadow removal Method B illustrated". Some results of the effectiveness of the shadow removal technique on frames of videos, which would have caused significant problems originally, are shown below (and overleaf):

Fig 2 – Shadow removal results (Non-simple background):

2(a) - Current frame of video

2(b) - Moving Pixels (no shadow removal)



2(c) - Moving Pixels (after shadow removal)

Fig 3 – Shadow removal results (plain background):

3(a) – Current frame of video 2

3(b) – Moving pixels (no shadows removed)

3(c) – Moving Pixels (after shadow removal)

3.4.12 – Dealing with occlusion (Proposed solution)

Although not deemed necessary for this application as the program is only interested in counting the number of passers-by in a video, it is an important part of a number of tracking algorithms and merits a brief discussion. All the necessary data structures and concepts are in place for dealing with the recognition of people during occlusion using two different methods. The first is based on converting an object's histogram into a probability distribution and using this distribution to determine which pixels in the group image belong to which individual object [**McKenna, Jabri, Duric & Wechsler**]. The probability conversion formula is:

$$\mathbf{P(x|}\boldsymbol{i}\mathbf{) = H}_{\boldsymbol{i}}\mathbf{(x) / A}_{\boldsymbol{i}}$$

where A_i is the area of the region histogrammed.
The subsequent reasoning for determining which person is which in a group is as follows:

For each person x in group G and for each pixel in the group's mask, *i*, the probability P(x|*i*) is obtained using *i*'s colour model:

$$\mathbf{P(}\boldsymbol{i}\mathbf{|x) = P(x|}\boldsymbol{i}\mathbf{)P(}\boldsymbol{i}\mathbf{) / } \sum_{\mathbf{j}\,\varepsilon\,\mathbf{G}} \mathbf{P(x|}\boldsymbol{j}\mathbf{)}$$

where,

$$\mathbf{P(i) = A}_{\boldsymbol{i}}\mathbf{ / } \sum_{\mathbf{j}\,\varepsilon\,\mathbf{G}} \mathbf{A}\boldsymbol{j}$$

Which is basically Bayes probability rule.

The second method is based on the colour indexing method from which histogram intersection (see 3.4.6) is derived [**Swain**]. "Experimentally Swain demonstrates that [the] histogram intersection method works well even when objects are partly occluded. This is to be expected, as a colour histogram is an accumulation of global evidence. The coloured areas not occluded should still be sufficiently discriminatory to allow correct object identification" [**Finlayson**].

Either method could be boot-strapped to the current algorithm if necessary as the histogram information is already available and in a form easily usable by both methods.

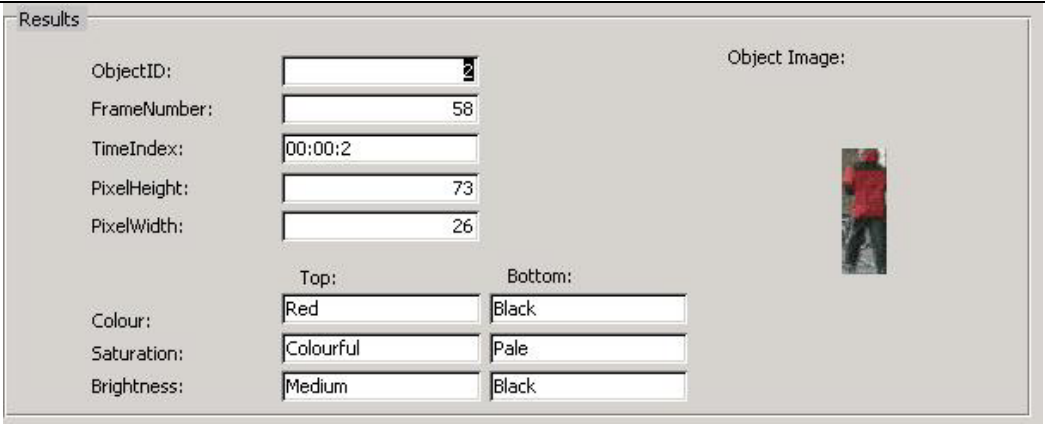
3.4.13 – The sample application

In order in illustrate one of the possible applications of a project of this nature (and indeed to implement a number of the projects proposed sub-goals) a sample database system was developed. The database stores descriptive information about the people seen walking in front of the camera, along with the frame number and time index of the sighting. Database connectivity functionality was added to the c++ tracking application along with a number of functions associated with describing a person.

As easy as describing someone may seem (e.g. “that person is wearing a green jumper and blue jeans”) this is actually not a simple task to replicate programmatically. In the above depiction of a typical human’s appearance a number of implicit concepts are used. What is green for example? What are Jeans? How blue are they? These are some of the technical details that need to be tackled in order to produce something that represents a “human-eye” view of the world from a user’s perspective.

The database’s user interface allows for the searching of the images produced by the tracking algorithm for people matching a particular description (see example overleaf).

Example of the database application front-end:



Screenshot of some of the results of the database applications search.

This example of the programs use should show the potential of the projects real-world use. If attributes such as a person's height, build, hair colour and sex could be added then an advanced digital surveillance system could be conceivably developed.

3.5 – Summary

In this chapter an introduction to the main stages of the development of the project was given and a general description of the problems, methods or algorithms used in the development of that stage were discussed. In the next section a more technically in-depth discussion of each stage described in this section will be given along with any relevant algorithmic implementations.

Chapter 4 – Action & Implemention

4.1) Introduction

In this section the actual implementations of the stages introduced in the chapter four are discussed. A more technical description of the algorithm used for each stage is provided along with any relevant pseudo-code. The full code will not be listed here but a complete listing of the project code is on the accompanying CD.

It must be noted that the generic pseudo-code given in this section will not match directly with the actual code of the project as much of the functionality of each algorithm was changed or tweaked in order for the application to run correctly and efficiently. Also, much of the code associated with each stage, whilst developed independently, has since been integrated together. Many of the data structures used in the code are not mentioned here and the final implementation of the system looks quite different from the pseudo-code in this chapter. The pseudo-code is simply there to help the reader understand the appropriate algorithm in its simplest form, and it should give a broad idea as to how each stage was approached.

Section 4.3 will introduce the main object classes created and give a brief discussion of their use

4.2) – Implementation of algorithm stages

4.2.1 – Finding Moving Pixels and the adaptive background model

The basic algorithm behind background subtraction is to search through each frame in the video file row by row. For each pixel in the row, the difference between the RGB values of the pixel in the current frame and the background image for this pixel is checked. If the difference in any channel is greater than a certain threshold then the pixel in considered to be moving.

In order to make the algorithm more general and forego the need for a hard-coded threshold value, an adaptive background model is used. An adaptive background model stores the average value, and the standard deviation from this mean, for each background pixel (this mean is updated for each frame the pixel is determined to be background). Instead of checking to see if the difference between the current frame's pixel at this position is greater than some pre-defined value, the difference between the current pixel and the mean value for the background at this position is checked. The threshold used now becomes 3 times the standard deviation at this position. (The standard deviation has a minimum value of a pre-determined standard deviation associated with the camera).

Algorithm:

```
For each row(r) in the image
        For each column(c) in the image
                If |image(r,c) – mean(r,c)| > 3 * stdDev(r,c) then
                        Pixel is moving pixel
                Else
                        Pixel is background
                        Update model
                End
        Next Column
Next Row
```

4.2.2 – Noise Reduction

The purpose of the noise reduction algorithm is to eliminate any of the random pixels mis-classified as moving pixels by the background subtraction algorithm. Each pixel in the binary image produced by the background subtraction (where the white pixels represent moving pixels and black ones are background) is taken and for each of these pixels the surrounding pixels are checked to see if they are also moving. This is done using a convolution mask that is contains the weighting associated with each position surrounding the current pixel (see section 3.4.2). The assumption behind this approach is that any moving object consists of a large number of connected moving pixels and so any single pixel, or small group of pixels can be safely ignored.

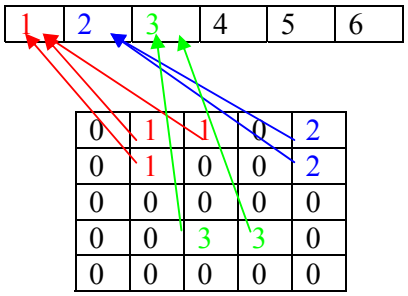
It is better to remove these pixels now in order to vastly reduce the number of regions produced by the connected component analysis for increased efficiency. A convolution mask is applied to each pixel as follows:

Algorithm:
//note since we are dealing with a binary image each pixel's value is either a 0 or 1 //in this case
For each pixel in the neighbourhood of the current pixel
Multiply this pixel value by the weight for this position //in convolution mask
Add this value to a total
Next pixel
Divide the total by the summed value of all the weights
If this value is greater than 1/2 the sum of the weights then pixel is part of a larger object
Else this pixel is noise

4.2.3 – Connected Component Analysis

The connected components analysis algorithm used in this project is an iterative version that uses a two-dimensional array of pointers. Each element of the array corresponds to a pixel in the image being analyzed. The pointer points to a non-zero label if the pixel is an object pixel (has been identified as moving by background subtraction) otherwise they point to a integer intialised to zero. The labels are stored in a separate one-dimensional array.

Example:



Where 1 in the two-dimensional array means a pointer to the value 1 etc (and 0 means no region associated – a pointer to an integer set to zero)

So that when equivalence is found only one value (in the one-dimensional array) has to be changed in order to change the label associated with a region. So for example in the above array in order to note the equivalence of regions 2 and 3, changing the 2's to 3's in the one-dimensional array is all that is required.

Algorithm:

```
For each pixel in the image
        If pixel is a moving pixel (i.e possibly part an object)
                If pixel has no label (pointer for this pixel in 2D array points to 0)
                        Get a new label (add element to Region Array)
                        Assign Label to pixel (point pointer to new label value)
                        Check the neighbouring pixels
                                For all unlabelled neighbours which are moving pixels
                                        Label with current label
                                For all differently labeled neighbours
                                        Note the equivalence
                If pixel has label
                                Check neighbouring pixels
                                        (as above)
next pixel
```

Noting equivalence merely iterates through the one-dimensional array and replaces the equivalent label with the current label. That way all the pointers in the 2D array that did point to the equivalent label value now all point to the new labels value. It is far more efficient to iterate through a 1-Dimensional array than a 2-Dimensional array, which makes this an effective method of processing equivalences (which is important because connected component analysis is a computationally expensive part of the overall algorithm and must be run every for every frame).

4.2.4 – Basic Tracking

Basic tracking (as described in 3.4.6) is crude but effective. It is implemented by taking each region that is found in the current frame being processed that is of sufficient size to be one or more people, comparing the centre point of that region to the centre point of each region currently being tracked and finding the tracked region with the closest distance between the centres. If this distance is smaller than a certain predefined distance (a small value itself since regions cannot move very far between frames) then these regions are the same and the tracked region is updated with its new values. Otherwise a new region has been identified for tracking.

```
Algorithm:

//SMALLEST_PERSON_SIZE – constant representing the smallest area in pixels of a region that
//could be one or more people
//SMALLEST_ALLOWED – constant small value representing the smallest distance allowed
//between centre points of the same region between two frames

For each region generated by connected component analysis
        If region area > SMALLEST_PERSON_SIZE
                If estimate of group size > 0 //check aspect ratios – see 5.2.8
                        Get region centre point (x,y)
                        For each region currently being tracked
                                Find the one with the smallest distance between the centres
                        End for
                        If this smallest distance < SMALLEST_ALLOWED
                                //region found
                                Update information about region (new centre etc)
                        Else
                                Generate a new region monitor and add to tracker
                        End if
                End if
        End if
Next
```

4.2.5 – Grouping People

Although the grouping of people is a relatively simple concept, its implementation has a number of difficulties. Two objects, which are separate in one frame, may come together in the next but if they do, there will only be one large region with a very different centre point and the two smaller regions will have effectively disappeared. At first a prediction algorithm was used to try and foresee two regions colliding and creating a new predicted region before it happened for tracking in the next frame. This had a number of significant drawbacks in that it was clumsy to code, required the generation of new regions based on estimations and suffered irreparably if a misclassification was made (i.e. if two objects didn't come together as predicted or if two objects did come together when not predicted). This sort of misclassification is possible given the fact that the morphological operations (described in section 3.4.3) will expand the areas of the moving object regions and may cause unforeseen connections between objects. This approach was discarded because of these failings and a new approach was created.

A new approach was needed that looked at the problem from a different angle – from the perspective of after a collision has occurred.

(Please note that the word "group" as used in the following explanation refers to an object of type ObjectGroup as defined in section 4.2 and may be, and frequently is, a single person or a "group of size 1". Irrespective of how many people or regions are contained within a "group", the group will have a single bounding region upon which the following reasoning is based. If the group does contain more than one region then the bounding region of the group will be the smallest rectangular region which encompasses all the regions inside the group, otherwise it will merely be the region of its single member).

If a region identified by connected components analysis could not be matched to a currently tracked group (i.e. either it is a new object entering the scene or it is the result of two or more tracked groups coming together) then the list of currently tracked groups is checked for any groups whose bounding region is at least partially

contained within the current region. If any are found then this region must be a group containing other groups and so each tracked group found during the check is added to this new object group. If no groups are found then this is simply a new object to be tracked. This works because any new region that appears, that has no corresponding region tracker, and partially contains other currently tracked groups, must have been formed from these tracked groups colliding.

Algorithm:

```
//new region (New_Region) has been found that doesn't match any currently
//available objects region.
//new group (New_Group) is therefore being created which is currently empty (i.e.
//holds no region monitors yet)

For each Group currently being tracked
        Check if current groups region border overlaps New_Region's border
        If so
                Add current group to New_Group
                Remove current group from tracked Groups list
        End if
Next
If no overlapping groups are found
        //completely new object entering the scene so initialise a new ObjectGroup
        Add New_Region to New_Group
End If
Add New_Group to tracked Groups list
```

4.2.6 – Generating Colour Histograms

Generating colour histograms is only done when a group being tracked has only one member (or at least only one region monitor object inside it). This means that colour appearance models are only built up for each individual object before it enters a group (or after it leaves).

A histogram object is represented by a three-dimensional array of integers (each dimension corresponds to a colour channel in RGB colour space). The function to generate a histogram uses a more sparsely quantised version of the input frame for the histogram calculation (the scene is reduced to 4 bit quantisation to increase efficiency). It takes a specific region monitor and for each pixel X within the border of the region being tracked checks to make sure that pixel X is part of the object being tracked, and not part of the background, and then increments that value (X=r,g,b) in the region monitor's histogram array. The end result is a 3D array with peaks at levels of high colour in the object (so a very red object might have very high values in its [16][x][y] bins and low values elsewhere – see note on the RGB colour space in Appendix B). The function also calculates the actual area of the region (as the border surrounding the region will include background pixels as well as object pixels) and this area will be used later for the histogram intersection method.

Algorithm:

```
//curr_frame_4bit – the current video frame in 4-bit quantisation
//this_region_monitor – the current region monitor object whose region we are
//histogramming
//regions_image – the regions produced by connected components analysis

Re-initialise this_region_monitor's histogram array to zeroes
Obtain a pointer to this_region_monitor's histogram for updating
For each row(r) within this_region_monitor's border (height)
        For each column(c) within this_region_monitor's border (width)
                If pixel(r,c) is also in the regions_image with this_region_monitor's label
                //(i.e.) if this pixel is part of the region and not part of the background
                        Increment histogram bin at x=(r,g,b) where r,g,b are the values of
                        the R,G and B channels of the pixel(r,c)

                        Increment the actual object area of the region
                End if
        Next
Next
Store the actual object area
```

4.2.7 – Using Histograms to aid tracking

Histograms are used to aid tracking when two or more people, who have been tracked individually already, come together into a group of people (a "multi-group" – or a group object that contains more than one person) with a single region and then subsequently split apart again. The tracking algorithm must be able to ascertain which new region (created by the split in this multi-group's single region) corresponds to which member of the group. This is done using histogram intersection. If a region is found with no matching object in the list of groups being tracked, and is partially touching an existing multi-group, then this region is assumed to be a member of the group splitting from the main group. As such, each member has its histogram array compared to the new regions histogram using the histogram intersection method and a probability that these histograms match is calculated. The member of the group with the highest probability of matching is assumed to represent the new region and is removed from the multi-group and put into a new group of its own to be tracked separately.

The basic formula for the histogram intersection method is:

$$H1 \cap H2 = \sum_i \sum_j \sum_k \min(H1(i,j,k), H2(i,j,k))$$

with the result divided by the area of the model histogram.

(Note: This approach works well for most cases, but cannot rigorously cope with all the possible permutations of people and groups interacting, as there are many ways in which the various occlusions can become too complex to handle. These cases occur infrequently, so the algorithms results are not significantly affected.)

Algorithm:

```
If new region found with no corresponding group currently being tracked
        If this region overlaps a currently tracked multi-group (group G)
                //region is a splitter from this group
                Histogram this new region (H1)
                For each region monitor (RM) inside group G
                        Let H2 be the histogram for RM
                        ProbabilitySameRegion = HistogramIntersection(H1,H2)
                Next
                Region monitor with highest probability is matched
                Remove this region monitor from the group*
                Add this region monitor to a new group
                Add this new group to the list of tracked groups
        Else
                Region has simply just entered scene
        End if
End if
```

*** This is not quite accurate because if you remove a member from a group of size 2 the remaining group will no longer classify as a multi-group so when the second "splitter" region is found it will not be close to a multi-group anymore and so will not be matched correctly. A more complicated approach is actually taken but the basics of the algorithm can be more easily understood this way.**

4.2.8 – Estimation of group size

The group size estimation function is basically a classification function that takes a region, calculates various statistics on it and, based on the results, returns an approximation of the group's size. It uses constant upper and lower bounds on single, double and triple group aspect and object ratios to classify the object count of the region. As these constants represent ratios, they are the same irrespective of the video file being processed, how far the object is from the camera or the size of the object.

Algorithm:

```
//X_PERSON_ASP_RATIO_MIN, X_PERSON_ASP_RATIO_MAX
// where X is ONE, TWO or THREE are the constant upper and lower bounds on the
aspect ratio of a single person (X=ONE) etc.

get aspect ratio value of region (Aspect_Ratio)
get object ratio value of region (Object_Ratio)
if Aspect_ratio > ONE_PERSON_ASP_RATIO_MIN and < ONE…MAX then
        if Object_Ratio < ONE_PERSON_OBJ_RATIO_MIN and < ONE…MAX then
                Group is of size 1
        End if
Else if Aspect_ratio > TWO_PERSON_ASP_RATIO_MIN and …
        if Object_Ratio < TWO_PERSON_OBJ_RATIO_MIN and…
                Group is of size 2
Else if Aspect_ratio > THREE_PERSON … and < …
        If Object_Ratio > THREE_PERSON and < …
                Group is of size 3
Else
        Group is of size 0 //does not match a person of group of persons
End If
```

4.2.9 – Shadow removal

As was seen in section 3.3.9, the shadow removal technique used in this project is divided into two different sets of rules depending on the type of background pixel. If the background pixel is plain, i.e. there not a large difference between the background pixels saturation and value channels in HSV space, then method (a) below is used. If the background pixel has a substantial difference between its S and V channels method (b) is used.

Method (a):

```
//back_image = background image
//curr_image = current frame image
For each row(r) in the curr_image
        For each column(c) in the curr_image
                BackDiff = back_image(r,c) S Channel – back_image(r,c) V Channel
                CurrDiff = curr_image(r,c) S Channel – curr_image(r,c) V Channel

                Hue_Diff = | curr_image(r,c) H channel – back_image(r,c) H channel |
                If | BackDiff – CurrDiff | > threshold1 or HueDiff > threshold2 then
                        Pixel is moving and is not shadow
                Else
                        Pixel is either not moving or is shadow
                End If
        Next
Next
```

Method (b):

```
//back_image = background image
//curr_image = current frame image
//THRESHOLD is chosen to be 60% of the value of the magnitude of the difference
//between the S and V values of a the background pixel
For each row(r) in the curr_image
        For each column(c) in the curr_image
                SVDiff = | back_image(r,c) S channel – curr_image(r,c) V channel |
                VSDiff = | back_image(r,c) V channel – curr_image(r,c) S channel |

                If SVDiff > THRESHOLD and VSDiff > THRESHOLD
                        Pixel is either background or shadow
                Else
                        Pixel is a moving pixel
                End if
        Next
Next
```

Method (a) above is conceptually similar to the original moving pixel calculation explained in section 3.3.1, with the magnitude of the difference between S and V channels used as a cue instead of each channel individually, and it is easy enough to see how and why the method works. Method (b) however is quite unusual and is a little obscure. To see how and why this method works in detail please refer to Appendix A "Shadow Removal Method B Illustrated" for an example of the method in action.

4.2.10 – Sample Application

The sample application is a database application that uses a Microsoft SQL Server 7.0 back-end database with a Microsoft Access 2000 VBA based front-end. The connections to the database are ADO based between Access and SQL Server and ODBC based between the c++ algorithm and the SQL Server. The technical details of the connection architectures used are not really relevant here and instead the approach taken to "describing" someone will be discussed briefly.

In order to describe a person's appearance, the HSV colour space was used, as this method of defining colour is more intuitively "human". The Hue channel is used to determine what colour something is (see picture below), the level of how colourful a region is can be found by measuring the Saturation channel and the Value channel can be used to calculate how bright or dark the colour is (see Appendix B for more details on HSV space).

HSV colour space - the hue channel:

In this way the code for establishing a accurate portrayal of a persons appearance counts the number of pixels of each colour in each half (vertically) of the object and the one with the highest number of pixels is said to be the principle object colour (as such the code essentially only deals with homogenous regions of colour or predominant colours).

The colours the program can classify are: Red, Orange, Yellow, Green, Cyan, Blue, Pink/Purple, Brown, White and black (the last two are not actually colours from the programs perspective and are determined differently but stored in the database in the same way as the others – see Appendix B on HSV space for details). The saturation was divided into: Dull, Pale, Medium, Colourful, Very Colourful and Brightness into: Black, Dark, Medium, Light and White. Each of these descriptions has a constant value associated with it and a classification function was written, which takes a pixel's channel values and returns the corresponding colour code, saturation code and brightness code.

Algorithm:

```
//ColourValues[] – an array representing the count of each colour in the object
//SaturationValues[] – an array representing the count of the different saturations of
//the object
//BrightnessValues[] – same as above but for brightness levels
//Get pixel colour refers to a function which classifies the Hue channel into a specific
//colour code, similar for saturation and brightness

For each row (r) in the region being described
        For each column (c) in the region
                Get pixel(r,c) colour code Colour_Code
                Get pixel(r,c) saturation code Saturation_Code
                Get pixel(r,c) brightness code Brightness_Code

                Increment ColourValues at index Colour_Code
                Increment SaturationValues at index Saturation_Code
                Increment BrightnessValues at index Brightness_Code

        Next
Next

Find largest value in colour array and store as dominant object colour
Same for Saturation and Brightness arrays
```

4.3) Classes and Data structures

The two main classes used by the program are RegionMonitor and ObjectGroup. A region monitor holds all the basic attributes of a particular region, produced by connected components analysis (see 3.3.4), which is deemed *trackable* (see basic tracking 4.2.4). For each frame where a region matching a currently held RegionMonitor, is found, the RegionMonitor in memory is updated with its new attributes.

An ObjectGroup represents a group of RegionMonitors. In its basic form an ObjectGroup holds just one RegionMonitor and is a "Group of size 1" and is simply a container for the RegionMonitor. If two regions collide and become one single region then their respective RegionMonitors will be put inside the same ObjectGroup (an ObjectGroup "carries" RegionMonitors by placing them within an internal linked list). An ObjectGroup has its own private RegionMonitor as well as any it is "carrying" so that when two or more RegionMonitors are in the same group, the overall bounding region can be tracked separately until the regions split apart again. When the regions split apart again the separate region monitors are removed from the current ObjectGroup and put into a new ObjectGroup on their own and this new group is added into the list of ObjectGroups being tracked.

The program maintains the temporally consistent list of ObjectGroups in an MFC CList structure (a form of linked list) and tracks each by matching the regions found in the current frame to the bounding regions of the ObjectGroups held (see basic tracking 4.2.4).

4.4) Summary

This section looked more comprehensively at the technical side of each stage of the project. The general pseudo-code for each major algorithm was given, and some of the procedural difficulties arising at the various stages were discussed. This section was written to give the reader a general idea as to how each stage was handled along with an understanding of any complications that arose during implementation.

Chapter 5 – Review

5.1) What Went Well & Why

The project was a success for the most part, in the end being able to reliably classify and track people, groups of people as a single object and groups of people as multiple objects. The main goals of the project were achieved and some of the possible sub-goals implemented. The tracking algorithm is fairly robust and yet uncomplicated enough to be computationally efficient. It is also quite expandable to more complicated tasks if necessary as shown by the proposed occlusion resolution algorithm in section 3.4.12. The shadow removal technique works very well and although it took some time to find a working solution, in the end was very necessary for the rest of the program to work correctly. As the sample application shows, the program has interesting potential in a number of areas such as surveillance. All in all the application works well within certain constraints and, with some further development, could be scaled to a workable solution to some real-world problems.

5.2) What could be improved & how

There are two areas where the program could be significantly improved. The first is in additions to the algorithm to cope with certain complicated situations that arise. The second is in additions to the functionality of the algorithm in order to add extra information associated with each person in the scene.

Algorithm additions:

5.2.1 – Complex backgrounds

The current algorithm has certain problems with complicated backgrounds. For example, if there is an object in the background, which is the same colour as the clothes worn by someone who walks past it, it will cause the misclassification of moving pixels as background. The problem is reduced as much as possible in this project by the adaptive background model, which helps reduce the threshold needed for moving pixel classification dynamically frame by frame, and the morphological closing operations to try and fill in the remaining gaps. These concepts are not a direct tackling of the problem, however, and merely serve to reduce the effects of it and in cases where the problem occurs severely they have little effect (see example overleaf). A future enhancement to the program would deal more effectively with this issue directly. There is, however, no easy way of dealing with this problem (how can a pixel be classified as moving if it looks exactly like the corresponding background pixel?). One suggested way is to have two separate adaptive background models [**McKenna, Jabri, Duric & Wechsler**], one based on chromaticity and one Sobel edge detection and combine the results to form moving pixels but that is beyond the scope of this project to cover.

Example of background difficulties:

Background Image of scene

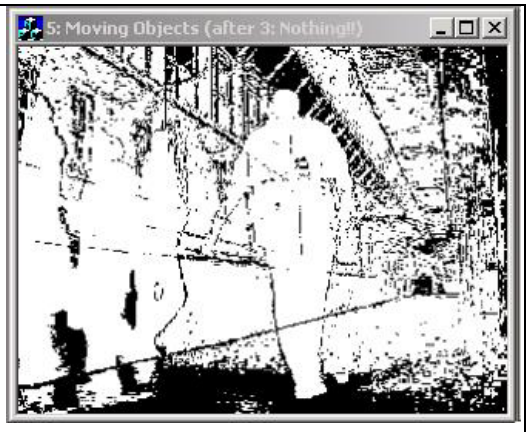
Current Frame (note the colour of the lady moving left matches closely the colour of the bin she obscures)

This is the resulting moving pixels image- the bin outline can be clearly seen here as the program thinks none of those pixels are moving.

5.2.2 – Rapidly changing illumination

A second algorithmic problem, which only occurs in certain specific circumstances, is caused by a rapidly changing illumination occurring across the whole scene. This sort of thing happens when, for example, a light switch is turned on or the sun moves out from behind a cloud. It causes significant problems to the moving pixel detection algortithm because every pixel is changed significantly. The problem is reduced as a by-product of the shadow removal algorithm (as shadows are effectively the inverse effect of increase illumination) but not enough to prevent severe degradation in moving pixel determination for a large sudden increase in illumination (see example below). An improvement to the project might look at this problem more explicitly.

Example of illumination problem (sunlight):

	
Frame 0 – some moving objects clearly defined from the backgound	*About 30 frames (or 1 second) later and the changing illumination has caused big problems for the moving pixel generation*

5.2.3 – Large group size estimation

The final algorithmic problem is in the estimation of group size and the way in which large mixed groups of people are handled. If a large group (maybe 5-10 people clumped together) enters the scene, it is very difficult to estimate that groups' size using the current methods. Since a group like this will probably not be walking together side by side the aspect ratio will probably only be able to tell you that there are more than three people in the group. Since there can be no use of region area on the determination of group size (a group of size two can have as small an area as one person if seen from a greater distance) there are no easy geometric cues to aid in group-size estimation. A further development to the application would be to use a more sophisticated estimation function, possible based on face extraction or on peaks (or bumps) at the top of a connected region (each head in the group might have a corresponding peak in the identified region).

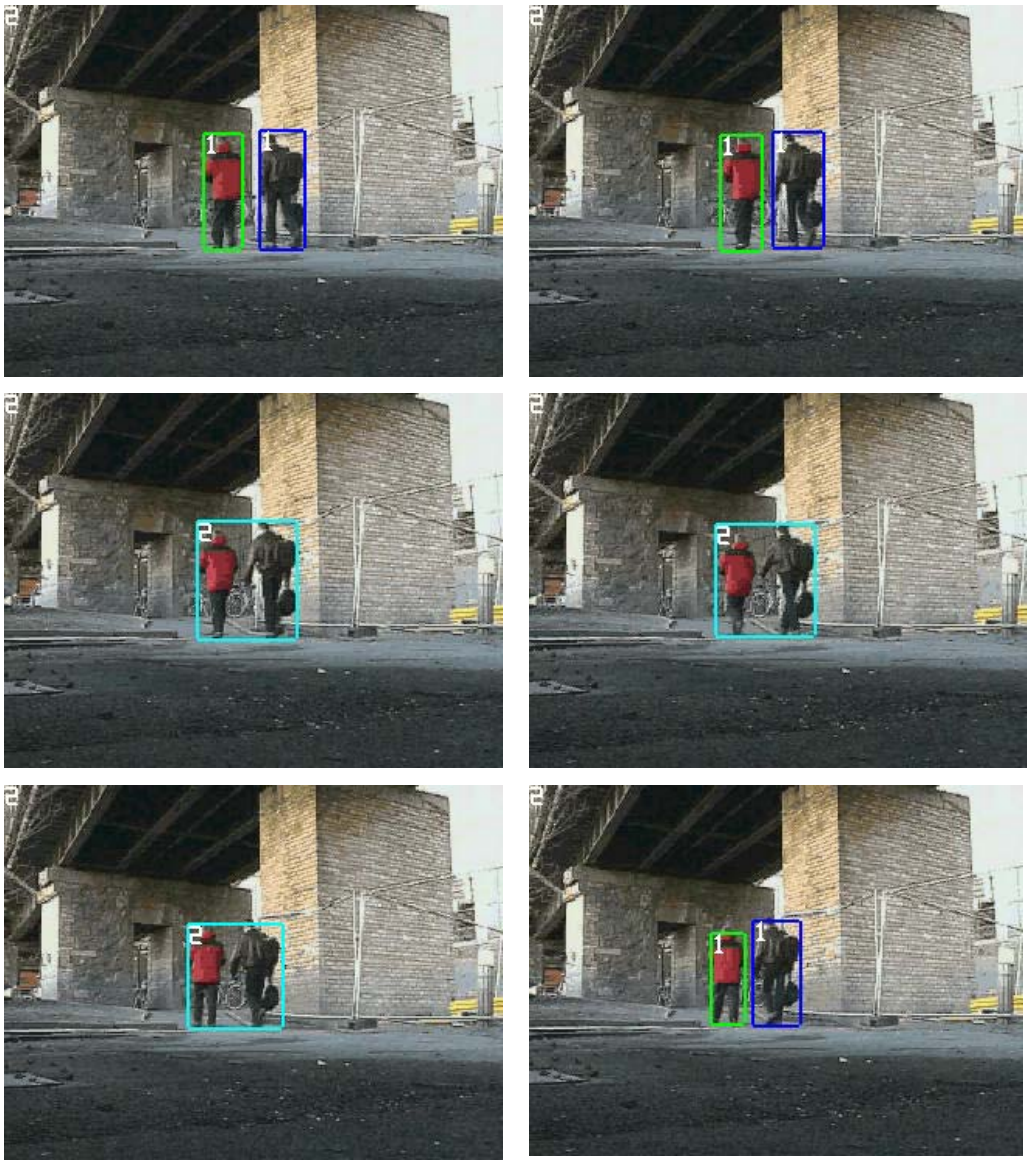
Functionality enhancements:

5.3.4 – Additional object information

A possible functionality addition would be more descriptive information about people who walk through the scene. For example a person's height, build, hair colour or sex might be added to have more cues to search from in the database application and more real-world significance from a security surveillance systems point of view. Or more complicated reasoning information could be stored such as whether a person was running or not (for security purposes) or whether a person stopped at any point in the scene for a certain amount of time (for marketing purposes – how many people stopped to look at a billboard for example). Unfortunately, these descriptors were beyond the goals of this project – but would certainly be welcome additions from an applications perspective.

Chapter 6: Sample Results

The following are some frames (every second consecutive frame) from a sample video file upon which the full algorithm has been run. It can be seen that initially the individual people are successfully identified and tracked frame to frame. When they come together as a group they are tracked as two people until they split apart again where they are recognised correctly and tracked individually again. For a more detailed sample please see the sample movie files (in .AVI format) on the CD accompanying this report.



Chapter 7: References

Papers & Journals:

[Adiv] : *G. Adiv – " Determining Three-Dimensional motion and Structure from optical flow generated by several moving objects" – IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 7, July 1985*

[**AElgammel, DHarwood, LDavis] :** *Ahmed Elgammal, David Harwood, Larry Davis "Non-parametric Model for Background Subtraction" - 6th European Conference on Computer Vision. Dublin, Ireland, June/July 2000.*

[**Aggarwal & Cai**] : *J.K. Aggarwal, Q. Cai, "Human Motion Analysis: A Review," Computer Vision and Image Understanding, 73:428--440, 1999.*

[Blake, Curwen & Zisserman] : *A. Blake, R Curwen, and A. Zisserman – "A framework for spatio-temporal control in the tracking of visual contours" – International journal of Computer Vision, Oct 1993*

[**Feiguth & Terzopoulos**] : *Fieguth, P. and D. Terzopolous, "Colour-Based Tracking of Heads and Other Mobile Objects at video Frame Rates", CVPR-97, 1997*

[Finlayson] : *Graham David Finlayson "Colour Object Recognition" – Masters Thesis University of Strathclyde, Dept of Computer Science.*

[Gennery] : *D.B. Gennery "Tracking Known Three-dimensional objects" – Proceedings American Association on Artificial Intelligence, 1982*

[Hallam] : *J. Hallam – "Resolving Observer Motion by Object Tracking" – Proceedings International Joint Conference on Artificial Intelligence, 1983*

[**Khan & Shah**]: *"Tracking People in the Presence of Occlusion" - Sohaib Khan and Mubarak Shah, Computer Vision Lab, School of computer science, University of central Florida.*

[Kohler] : *Markus Kohler – "Using the Kalman Filter to Track Human Interactive Motion – Modelling and Initialisation of the Kalman filter for Translational motion" – Universitat Dortmund, Informatik VII, D-44221 Dortmund, Germany*

[**Lawrence, Giles, Tsoi & Back**] : *Steve Lawrence, C. Lee Giles, Ah Chung Tsoi, and Andrew D. Back – "Face Recognition: a convolutional neural network approach" - IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition, Volume 8, Number 1 pp. 98-113, 1997.*

[Lemon, Rhodus & Hall] : *J.R. Lemon, W.T. Rhodus and F.R. Hall – "NEURAL NETWORKS AND THEIR APPLICATION", Departments of Entomology, Horticulture and Agricultural Economics and Rural Sociology, The Ohio State University*

[Matteuci] : *P. Matteucci, C.S. Regazzoni and G.L. Foresti – "Real-time approach to 3D object tracking in complex scenes" – Electronics Letters Vol 30. pp 475-477, March 1994*

[**McKenna, Jabri, Duric & Wechsler**] : *"Tracking Interacting People" – Stephen J McKenna, Department of Applied Computing, University of Dundee & Sumer Jabri, Zoran Duric, Harry Wechsler, Department of computer science, George Mason University, Fairfax, VA*

[Nickels] : *K.M. Nickels – "Model-based tracking of Articulated Objects" – PhD Thesis , University of Illinois, Urbana, IL, 61821, October 1998*

[**Porte**r] : *N.D. Porter "Neural Networks for the classification of facial features" – Department of engineering, University of Warick, Coventry.*

[**Raja, McKenna and Gong**] : *Y. Raja, S. J. McKenna, and S. Gong. "Tracking and segmenting people in varying lighting conditions using colour." - IEEE International Conference on Face & Gesture Recognition, pages 228-233, Nara, Japan, 1998.*

[**Rowley, Baluja & Kanade**] : *Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. "Neural Network based face detection" - IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 20, number 1, pages 23-38, January 1998.*

[Stark & Fuchs] : *K. Stark & S Fuchs – " A method for tracking the pose of known 3D objects based on Active Contour models" – Proceedings International Conference on Pattern Recognition, 1996*

[Swain] : *M.J. Swain "Colour Indexing (PHD Thesis)" – University of Rochester, Dept of Computer Science, 1990*

[Treglown] : *M. Treglown – "Lifecycle Models", University of Nottingham, School of computer science and technology.*

Books:

[Sonka] : *M.Sonka, V. Hlavac, R. Boyle – "Image Processing, Analysis, and Machine vision" – PWS publishing, Second Edition, 1999.*

Web:

[**BriefBook**] : *"The data analysis BriefBook" – An Extended Glossary* (http://rkb.home.cern.ch/rkb/titleA.htm)

[**Cardini**] : *Darren Cardini – "Adventures in HSV Space"* (http://www.beuna.com/articles/hsvspace.pdf)

[**Citysync**] : *Citysync ltd* (http://www.citysync.co.uk/)

[**Coverdale**] : *Coverdale Management Consultancy* (http://www.coverdale.co.uk/)

[**Edinburgh**] : *Edinburgh Online Graphics Dictionary* (http://www.dai.ed.ac.uk/homes/rbf/grdict.htm)

[**Janne**] : *Dr. Janne Heikkilä, Department of Electrical Engineering, University of Oulu, Finland*. (http://www.ee.oulu.fi/~jth/monitor/Monitor.html)

[Hjelmås, Lerøy, Johansen] : *Erik Hjelmås, Christel Beate Lerøy and Henry Johansen, Department of Electrical Engineering and Science Gjøvik College, "ICI System: a first attempt"* http://ansatte.hig.no/~erikh/papers/hig98_6/higrapp1.html

[**Minnesota & Honeywell**] : *Nikolaos P. Papanikolopoulos, Principal Investigator Department of Computer Science and Engineering University of Minnesota* http://www.dot.state.mn.us/guidestar/hovlanecount.html

Appendix A – "Shadow Removal Method B Illustrated"

This appendix gives a brief example of the shadow removal method B in action and tries to explain how this method works. As stated this method is applied to current frame pixels whose corresponding pixel in the background has a high difference between its Saturation (S) channel and Value (V) channel in HSV space. (This usually means a low saturation and high value as it would be unusual to have an everyday background with a very dark intensity yet very high colour factor).
In this example method (a) will be removed from the algorithm completely and scene where only method (b) is necessary will be used.

Method (b) Example:

<table>
<tr><td>
</td><td>
</td></tr>
<tr><td>Fig 1: Current frame</td><td>Fig 2: normal moving pixels generation</td></tr>
</table>

From the above figures it can be seen that there is a substantial shadow falling on the brick wall to the right of the two walking people. Now let us look at the VS and SV differences for each pixel in the image compared to its corresponding background pixel:

Fig 3: The VS differences: white pixels correspond to pixels representing background or shadows

Fig 4: The SV differences: very similar to the VS differences with some exceptions

From above it is clear that any background pixel that has a high difference between its own S and V channels, and is still a background pixel in the current frame, is represented as a white pixel (note the black pixels in the above that rightly correspond to background pixels will be taken care of by the method (a)). These two images are logically "AND-ed" together and then inverted (to convert from "white pixels are background" to "white pixels are moving objects") which is then combined with the original moving pixel information to get the following:

*Fig 5: VSDiff **and** SVDiff: white pixels now represent background or shadows*

Fig 6: Inverted and and-ed with the original image. It can be seen that the shadow has been successfully removed

Also, when two people are moving through a background area that has a high S-V difference none of their own object pixels are distorted because their S channel value

is brought close enough to the backgrounds V channel to (or vice versa) for the pixel to be classified as “not background” (see example below).

Fig7: VS Difference	*Fig 8: SV Difference*	*Fig 9: VS **and** SV difference*

Appendix B – "Some Vision Concepts Explained"

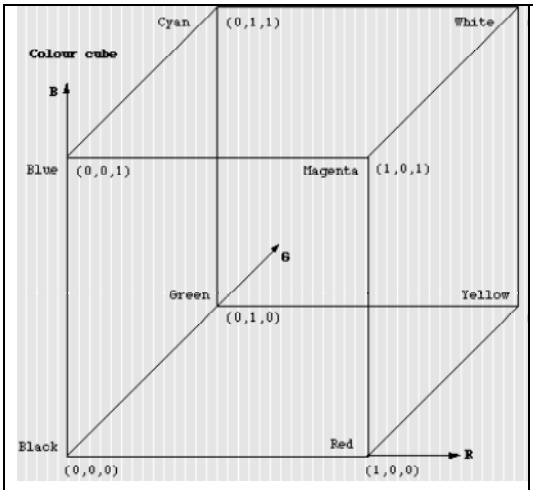
The following are some of the concepts referred to in this report, which may not be mainstream knowledge for people who are not familiar with the area of computer vision.

Colour spaces:

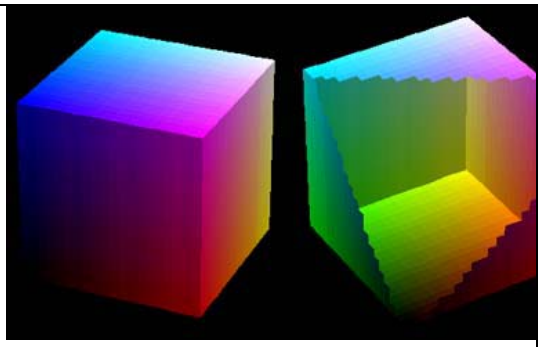
RGB:

The RGB ("Red", "Green" & "Blue") colour model describes a colour as a positive combination of three appropriately defined red, green and blue primaries. If the r, g and b components are defined as scalars constrained to a value between 0 (no intensity) and 1 (maximum intensity) all the definable colours will be bounded by a cube and it is typical to describe RGB combinations as co-ordinates on the cube (r, g, b). For example pure red is (1, 0, 0) and the secondary colour cyan is (0, 1, 1); darker colours have values closer to (0, 0, 0) (black) and lighter colours have values closer to (1, 1, 1) (pure white) [**Edinburgh**].

The RGB Cube:



The typical RGB cube

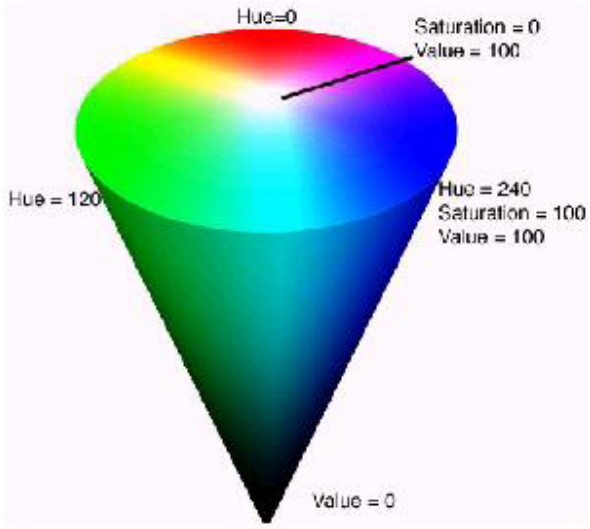


A 3D version of the RGB cube

HSV:

Most operating systems, image processing programs and texts treat images as collections of pixels comprised of red, green and blue values. This is very convenient for display purposes, since computer monitors output colour by combining different amounts of red, green and blue. However, most users don't think of colour in these terms. They tend to think about colour the same way they perceive it - in terms of hue (the English name we give colours, like "reddish" or "greenish"), purity (pastels are "washed out", saturated colours are "vibrant"), and brightness (a stop sign is "bright" red, a glass of wine is "dark" red). HSV space is what is known as a perceptual colour space because is models this human view off the world. Conceptually, the HSV colour space is a cone (see overleaf). Viewed from the circular side of the cone, the hues are represented by the angle of each colour in the cone relative to the 0° line, which is traditionally assigned to be red. The saturation is represented as the distance from the centre of the circle. Highly saturated colours are on the outer edge of the cone, whereas grey tones (which have no saturation) are at the very centre. The brightness is determined by the colours vertical position in the cone. At the pointy end of the cone there is no brightness so all colours are black. At the fat end of the cone are the brightest colours [**Cardini**].

HSV Colour cone:



Terms & Concepts:

Occlusion:

Visual obstruction. An occlusion occurs when an opaque surface prevents another surface from being seen. [**Edinburgh**]

Quantisation:

In its original meaning, quantisation is the step of passing from a continuous to a discrete variable, like in analogue-to-digital signal conversion. More generally, the term can be used for any method decreasing the precision of representation by eliminating part of the information [**BriefBook**]. In terms reducing image quantisation, an 8bit image has 8 bits per channel but frquently not all these bits are needed or used to represent the image fully and so the image can be just as easily represented with 6 or 4 bits per channel.