Background Subtraction Algorithms for a Video based system

by

Barton Profitt

Thesis presented in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

at the Stellenbosch University

Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa.

Supervisors:

Dr. K.M. Hunter
Applied Mathematics
Departement of Mathematical Sciences
University of Stellenbosch

Prof. B.M Herbst
Applied Mathematics
Department of Mathematical Sciences
University of Stellenbosch

2009

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, and that I am the owner of the copyright thereof (unless to the extend explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
B. Profitt

Date: .....

Abstract

Background Subtraction Algorithms for a Video based system

B. Profitt

Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa.

Thesis: MScEng (Applied Mathematics)

2009

To reliably classify parts of an image sequence as foreground or background is an important part of many computer vision systems, such as video surveillance, tracking and robotics. It can also be important in applications where bandwidth is the limiting factor, such as video conferencing.

Independent foreground motion is an attractive source of information for this task, and with static cameras, background subtraction is a particularly popular type of approach. The idea behind background subtraction is to compare the current image with a reference image of the background, and from there decide on a pixel by pixel basis, what is foreground and what is background by observing the changes in the pixel sequence.

The problem is to get the useful reference image, especially when large parts of the background are occluded by moving/stationary foreground objects; i.e. some parts of the background are never seen.

In this thesis four algorithms are reviewed that segment an image sequence into foreground and background components with varying degrees of success that can be measured on speed, comparative accuracy and/or memory requirements. These measures can be then effectively used to decide the application scope of the individual algorithms.

Uittreksel

Agtergrond aftrekkings Algoritmes vir 'n Video baseerde sisteem

("Background Subtraction Algorithms for a Video based system")

B. Profitt

Departement Wiskundige Wetenskappe,
Universiteit van Stellenbosch,
Privaatsak X1, 7602 Matieland, Suid-Afrika

Tesis: MScIng (Toegepaste Wiskunde)

2009

Om betroubaar dele van 'n beeld reeks te klassifiseer as voorgrond of agtergrond is 'n belangrike deel van baie rekenaarvisie sisteme, byvoorbeeld video bewaking, volging en robotika. Dit kan ook belangrik wees in toepassings waar bandwydte die beperkende faktor is, byvoorbeeld video konferensie gesprekke.

Onafhanklik voorgrond beweging is 'n aantreklike bron van informasie vir hierdie taak, en met statiese kameras, is agtergrond aftrekking 'n populêre benadering. Die idee agter agtergrond aftrekking is om die huidige beeld met 'n naslaan beeld van die agtergrond te vergelyk, en daarvandaan besluit op 'n piksel-na-piksel basis, wat is voorgrond en wat is agtergrond deur die observasies van die veranderinge in die piksel-reeks.

Die probleem is om die naslaan beeld te kry om mee te werk, veral wanneer groot dele van die agtergrond onsigbaar bly as gevolg van bewegende of stilstaande voorgrond objekte en sommige dele van die agtergrond word dalk nooit gesien nie.

In hierdie tesis word vier algorithms ondersoek wat 'n beeld reeks segmenteer in respektiewe voorgrond en agtergrond komponente met wisselende grade van sukses wat gemeet kan word deur spoed, vergelykbare akkuraatheid en/of

geheu gebruik. Hierdie metings kan dan effektief gebruik word om die applikasie veld van die individuele algoritmes the bepaal.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations who contributed to this work:

My study leaders Dr. Karin Hunter and Prof. Ben Herbst for their patience, guidance, ideas and motivation
The Applied Mathematics Division of the Department of Mathematical Sciences
The NRF who provided funding for my post graduate studies
My wife Juan-Marie for all her support and motivation
My parents for their encouragement
My friends for the late night coffee sessions

Dedications

This thesis is dedicated to my wife

Contents

List of Figures

List of Tables

Nomenclature

AVI - Audio Visual Interleave
CCTV - Closed Ciruit Television
GMM - Gaussian Mixture Model
MOG - Mixture Of Gaussians
MPEG - Motion Pictures Expert Group
PCA - Principal Component Analysis
PDF - Probability Density Function
TAPPMOG - Time Adaptive Per-Pixel Mixture Of Gaussians
USB - Universal Serial Bus
CC - Chromacity Coordinates
HSV - Hue Saturation Value
HSL - Hue Saturation Luminance

Chapter 1

Introduction

Background subtraction is an important and fundamental part of many computer vision applications such as real-time tracking, video/traffic surveillance and in human-machine interaction.

Figure 1.1: Surveillance monitor room

The central idea behind this is to utilise the visual properties of a scene for building an appropriate representation of the background that can be used for the classification of a new observation as foreground or background. The information provided by this lower-level processing can be valuable cues when performing high-level object analysis tasks such as object detection, classification, tracking and event analysis. To reliably classify parts of an image sequence as foreground or background makes these tasks more accurate due to less noise and more information present in a specific region. A system that

can do this accurately can be very helpful in situations such as in Figure 1.1, allowing the operator to only focus on situations that require attention and not wasting time scanning for activity through multitudes of monitors.

The most intuitive way is to have a reference image of just the background and then concurrently subtract each new observation image from the reference image, leaving only the foreground objects. But this method is sensitive to noise, change in lighting, non-stationary background objects and occluded background, as the static background model cannot account for these invariances. Another difficulty is that one often cannot clear a scene to get a background image, for example trying to clear a busy town square is impractical just to get a static image that is only usable under certain specific restrictions.

Figure 1.2: Stanley the autonomous vehicle

Another case to consider is where an autonomous robot that must be able to perform reliable background detection as soon as possible after coming to a standstill — whether the robot is in a familiar environment or not — there is no possibility to obtain a background beforehand, everything must start from scratch. This problem was solved admirably by Stanley, the winning entry for autonomous vehicles in the DARPA challenge, seen in Figure 1.2 [1].

In our modern society, gated villages, housing estates and retirement homes are becoming more prevalent and demand a high level of security [2]. As such, video surveillance systems are also becoming more prevalent in our modern day society, as can be seen in Figure 1.3. These systems can typically have a large number of cameras that cover outdoor and indoor scenes. One does also not have an infinite amount of processing power available, so thus the need for computationally lite algorithms arise. Algorithms that are suitable should have a fast enough runtime, which we also cover in the thesis.

Figure 1.3: Entrance to a gated housing estate, with one camera that is part of a larger network of cameras

To deal with these problems one wants to use an adaptive background model to account for multi-modal backgrounds, and handle objects that are moving. In this thesis four different methods of constructing a background model are considered: eigenbackgrounds, mixture of Gaussians, optical flow and AdaBoost. The algorithms are implemented and evaluated against each other using video sequences with common background subtraction problems or issues.

1.1 Literature Review

Background subtraction is part of the field of study known as computer vision. Extensive research has been done in this field and as it is of huge interest to society with many applications in surveillance and security, robotics, autonomous vehicle navigation, sports analysis, etc.

Some common algortihms for background subtraction include

- AdaBoost
- Eigenbackgound
- Gaussian Mixture Models
- Kernel density estimation

- Median background
- Optical Flow

Here we discuss the most commonly used background subtraction methods relevant to this thesis for background modelling.

1.1.1 The Background Subtraction problem

In many image processing applications one is interested in only certain parts of the image. For example, when doing object classification one wants to concentrate on the part of the image containing the object and when doing tracking the focus is on the moving object. If in tracking one could accurately and easily distinguish between the moving object, the foreground, from the background, tracking as well as in object identification, becomes easier where once the background has been accurately identified, a lot of information can be gained. This can be likened to a 'simple' two class classification problem: Given enough data, can we segment an image in to one of two classes ; Foreground or Background.

A common technique is to perform foreground segmentation by background subtraction in case of a stationary camera. This works by subtracting concurrent frames from each other, the regions where the images do not match are classified as foreground.

As background subtraction is a far from perfect art, there are several problems one needs to take into account when developing background subtraction algorithms. The following common problems are identified by Toyama *et al.* [3] and Harville *et al* [4; 5]:

Camouflage

When a foreground object has the same texture or colour characteristics as the background, allowing the object to blend into the background.

Illumination changes

Illumination changes occur due to the time of day or other external light sources such as car headlights or outdoor lighting being turned on and off. Such changes can often be interpreted incorrectly as foreground or making the current background model obsolete. Stauffer and Grimson, [6], discusses the case where the illumination of the scene being modelled changes during the course of the day.

Initialization

With cluttered scenery it is often impossible to build a robust enough background model very quickly, one must normally wait for the algorithm to learn the background over time.

Moved objects

This is when part of the observed background is removed, such as moving a chair indoors, a sleeping person waking up and moving away and outdoors when parked cars drive away. Most techniques using adaptive background modelling techniques can overcome this problem [6; 7].

Non-static background objects

Oscillating movements of trees blowing in the wind, clouds moving in the sky and lights and monitors flickering indoors are some of the examples of non-static background objects. The wallflower algorithm [3] deals with some of these problems.

Occluded Background

This is linked with the initialization problem, as it is not possible to build a background of invisible regions.

Shadows and reflections

As foreground objects move across the background scene, they cast shadows that are easily interpreted as foreground although it should be classified as background.

In [8] Porikli and Tuzel propose a model to make background subtraction robust against shadows. This is done by splitting the colour information in an image into brightness and chromacity. They make the assumption that the chromacity stays almost constant and that brightness changes indicate whether a pixel is part of a shadow. The pixel can fit one of four categories: background, shaded background/shadow, highlighted background or foreground.

Davis *et al* presents a similar simplified model in [9].

Thresholding

A threshold is typically used to decide whether a region should be classified as foreground or background. What should pass and what should not? The problem is very dependent on noise and is commonly solved by statistical methods.

1.2 Methods

A brief overview of the four algorithms studied in this thesis are given here, for full implementation details, please refer to Chapter 3.

1.2.1 Eigenbackground

Eigenbackgrounds are formed from eigenvectors derived from the covariance matrix of the probability distribution of the high-dimensional vector space of possible backgrounds. Similar research has been done on classifying faces using eigen methods [10] and [11; 12] explores eigen methods coupled with other methods, allowing for higher level heuristic programming and better results.

Moving objects contribute less to the model, thus the N eigenvectors provide a robust model for the background, allowing for small movements such as moving foliage, lighting variation and weather variation, provided the N eigenvectors are updated often enough. This method is also less computationally expensive and yields good results according to [10].

1.2.2 Optical Flow

The objective of optical flow is to find the 2D motion field in an image sequence.

The goal is to compute an approximation to the 2D motion field, a projection of the 3D velocities of surface points onto the imaging surface, from spatiotemporal patterns of image intensity. Barron *et al* [13] discusses the problem of evaluating different methods as the 'correct' optical flow. The ground truth is only available in situations with synthetic sequences. Horn and Schunk [14] also introduces the importance of reliability measures.

One common problem with optical flow is the window or aperture problem: as the search area around each pixel is very small, one-dimensional features can only be tracked in the direction normal to the motion of the object. Other shortfalls of these algorithms are occlusion, illumination and numerical instabilities caused by the dependence on numerical differentiation which is noise sensitive. Ranchin *et al*, Galic *et al* and Aggerwal *et al* [15; 16; 17] attempts different approaches to overcome these problems by using clustering and adaptive algorithms.

Finally Lucas and Kanade [18] describes the pyramidal version of their method for calculating optical flow. This approach is motivated by the fact that calcu-

lating flows for each pixel is expensive, but using different levels, the resolution becomes coarser reducing the cost of computation. As we do not normally have any prior knowledge about the current scene, we are mostly interested in general flow [18].

1.2.3 Mixture of Gaussians (MOG)

To make the background model more robust, several statistical methods have been proposed as solutions to the background model problem. Most common methods work by modelling the background by using a multi-modal probability density function (pdf) for every pixel. The pdf itself is commonly created as a linear combination of Gaussian probability functions. This is a common method in machine learning and pattern recognition, known as mixture of Gaussians (MOG).

These methods are popular due to the fact that they address many of the background subtraction problems explained earlier in this chapter. The algorithm proposed by Grimson and Stauffer [6] was compared in the Wallflower test done by Microsoft Research, [3], and worked very well.

Friedman and Russell [7] present an algorithm that creates different models for each pixel, depending whether the pixel represents background, shadow or a moving object. The algorithm also allows for illumination changes, by being updated unsupervised by an expectation-maximization method.

Both of these algorithms make the assumption that the background is seen for a long period of time, however this is not the case in many practical scenarios. Zivkovic and Suter *et al* [19; 20] discuss the results of modifying the Mixture of Gaussians (MOG) with additional heuristics to help differentiate models for pixels. Other approaches such as discussed by Davis *et al* in [9], adds kernel density functions together with a selective update mechanism.
Dibos *et al* [15] uses texture based discriminant features in a MOG, making for a robust model. Cristani *et al* [21] develops a spatial time adaptive per-pixel MOG that considers zones of interest. The background is modeled in the chromatic uniform areas that exhibit stable behaviour, minimizing foreground camouflage.

1.2.4 AdaBoost

AdaBoosting is a general rule-of-thumb algorithm that uses a committee of algorithms called weak learners. On its own, the weak learner is only slightly better than 50/50, but a combined vote of confidence from the committee

of weak learners turns out to be very accurate, see Freund and Schapire [22]. The algorithm has been implemented as a facial locating application in [23; 24]. Belaroussi and Milgram [25] uses AdaBoost in a facial tracker context.

Not much research has been done on the viability of AdaBoost algorithms modelling the background, so one of the aims of this thesis is to explore that route.

1.2.5 Colour spaces and chromacity coordinates

The detection of shadows as foreground regions is a source of confusion for subsequent analysis. Colour information is usefull for suppressing shadows by separating colour information from lightness information [26]. Different colour spaces also have different advantages when performing background subtraction [27; 28; 29]. The HSV colour spaces separates a RGB image into its hue, saturation and value or intensity components, whereas the YCbCr splits a RGB image into Y′, that is the luma component and Cb and Cr are the blue-difference and red-difference chroma components. The prime is to distinguish the Y as luma and not luminance.

1.3 Summary

In this chapter we presented the problems faced by the most common methods in computer vision for background modeling. We briefly discussed eigenbackgrounds, optical flow, mixture of Gaussians and AdaBoost. We provided an overview of these methods and learnt from their characteristics that they are suitable for development of algorithms to suit our purpose of a robust, accurate and CPU inexpensive algorithm.

Note that the four different algorithms were implemented first in Python to test viability and later in C++ for efficiency reasons. The algorithms were then compared and evaluated using video sequences that have different problem areas.

We also test using chromacity coordinates, HSV and YCbCr colour spaces to try and suppress shadow information.

1.4 Outline

The organization of this thesis largely follows the order in which the work was done. Relevant literature concerning background subtraction problems and algorithms are reviewed in this chapter. The implementation of background subtraction algorithms are described in Chapter 2, colour spaces and other methods used to suppress shadows are discussed in Chapter 3. The relevant hardware is explained as well as the implementation issues are discussed together with software choices in Chapter 4. Test and results of the background subtraction algorithms are presented in Chapter 5 and in Chapter 6 the general problems and possible future extensions are discussed.

Chapter 2

Methods and Algorithms

2.1 Eigenbackgrounds

The idea behind eigenbackground modeling is to extract and represent all the relevant information from an image efficiently, using Principal Component Analysis (PCA) to reduce the dimensionality of the data. This allows us to find a linear subspace for efficiently representing the background model within the entire image space as shown in Figure 2.1.

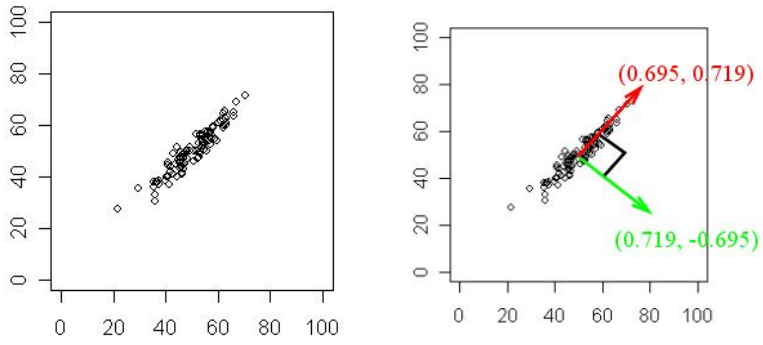


Figure 2.1: Principal Component Analysis example on a clustered data set

Moving objects contribute less to the model, thus the eigenbackground provides a model for the background, allowing for small movements such as moving foliage, lighting variation and weather variation, provided the eigenbackground is updated frequently enough.

Suppose you have data comprising a set of M frames of a video sequence to use for background training images, and you want to reduce the data so that

each observation can be described with only N variables, $N < M$. The data can be arranged as a set of data vectors $\mathbf{x}_1 ... \mathbf{x}_M$, where each $\mathbf{x}_j$ represents a single observation of the background. The mean $\mathbf{m}$ and the deviation from the mean $\mathbf{d}$ are computed

$$\mathbf{m} = \frac{1}{M} \sum_{j=1}^{M} \mathbf{x}_j, \tag{2.1}$$

$$\mathbf{d}_j = \mathbf{x}_j - \mathbf{m}. \tag{2.2}$$

Let $\mathbf{A} = [\mathbf{d}_1 ... \mathbf{d}_M]$, with covariance $\mathbf{C}$,

$$\mathbf{C} = \frac{1}{M} \mathbf{A}\mathbf{A}^T. \tag{2.3}$$

For background modelling purposes this covariance matrix gets very large making it infeasible to compute the eigenvectors and eigenvalues directly. For example with the image size as a standard 640 x 480 resolution, the size of the covariance matrix is $(640 * 480)^2 = (307200)^2$, which is very large.

Consider the eigenvectors $\mathbf{v_j}$ and eigenvalues $\omega_\mathbf{j}$ of $\mathbf{A}^T\mathbf{A}$ such that

$$\mathbf{A}^T\mathbf{A}\mathbf{v}_j = \omega_j \mathbf{v}_j. \tag{2.4}$$

Multiplying both sides by $\mathbf{A}$, we have

$$\mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{v}_j = \omega_j \mathbf{A}\mathbf{v}_j. \tag{2.5}$$

But this is an unstable way to calculate the eigenvectors, so we perform a Singular Value Decomposition of

$$\mathbf{A} = \frac{1}{\sqrt{M}} [\mathbf{d}_1 ... \mathbf{d}_M]. \tag{2.6}$$

Most software packages allows you to calculate the reduced form of the SVD, returning only the values that we are interested in.

The new image I is projected into the eigenbackgroundspace spanned by the N eigenvectors and is reconstructed as I'. Then it follows that $I - I'$ should render foreground objects as the eigenbackground subspace represents the largely static parts of the scene being modelled.

2.2 Optical Flow

The objective of optical flow is to find the 2D motion field between 2 successive images in a video stream.

Figure 2.2: Optical flow algorithm on an image

2.2.1 Lucas-Kanade method for computing optical flow

The idea of the algorithm is to calculate some function or velocity vector for each pixel in an image I. This function describes how quickly each particular pixel (p_x, p_y) in image I is moving across the image stream to image J along with the direction in which the pixel is moving. The aim is then to find the

displacement of a pixel (p_x, p_y), denoted by (d_x, d_y). This is estimated by minimizing the function E, defined as

$$E(d_x, d_y) = \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-w}^{p_y+w} (I(x,y) - J(x+d_x, y+d_y))^2 \tag{2.7}$$

for each pixel (p_x, p_y) in I. The size of the window or neighbourhood around the pixel where the displacement might be located is determined by w.

A pyramidal segmentation scheme [30], is implemented to make calculations more efficient and allow for large movement. This makes it easier to segment the faster moving foreground from the largely static background. The basic idea is to use results obtained from processing low resolution images as start values for the higher resolution images.

We call the original, high resolution image I_0 (the image at 0th level). From this the pyramid is computed where, I_{L+1} is at half the resolution of I_L. The same calculations are done at each level, starting by minimizing the function E by differentiating E with respect to $\mathbf{d} = [d_x, d_y]^T$ and setting the derivative equal to zero:

$$\begin{bmatrix} \frac{\partial E}{\partial d_x} \\ \\ \frac{\partial E}{\partial d_y} \end{bmatrix} = -2 \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-w}^{p_y+w} (I(x,y) - J(x+d_x, y+d_y)) \begin{bmatrix} \frac{\partial J}{\partial d_x} \\ \\ \frac{\partial J}{\partial d_y} \end{bmatrix}. \tag{2.8}$$

We also substitute the displaced image $J(x+d_x, y+d_y)$ with a Taylor expansion

$$\begin{bmatrix} \frac{\partial E}{\partial d_x} \\ \\ \frac{\partial E}{\partial d_y} \end{bmatrix} \approx -2 \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-w}^{p_y+w} \left(I(x,y) - J(x,y) - \begin{bmatrix} \frac{\partial J}{\partial d_x} & \frac{\partial J}{\partial d_y} \end{bmatrix} \begin{bmatrix} d_x \\ \\ d_y \end{bmatrix} \right) \begin{bmatrix} \frac{\partial J}{\partial d_x} \\ \\ \frac{\partial J}{\partial d_y} \end{bmatrix}. \tag{2.9}$$

Now we can interpret $I(x,y) - J(x,y)$ as the time derivative and denote it as $\delta I(x,y)$. The vector $\begin{bmatrix} \frac{\partial J}{\partial d_x} & \frac{\partial J}{\partial d_y} \end{bmatrix}^T$ is the image gradient and denote it as $\nabla I(x,y) = [I_x(x,y), I_y(x,y)]^T$. The gradient can be computed from the initial image I by a central difference operator:

$$I_x(x,y) = \frac{I(x+1,y) - I(x-1,y)}{2}, \tag{2.10}$$

$$I_y(x,y) = \frac{I(x,y+1) - I(x,y-1)}{2}. \tag{2.11}$$

Now we have

$$\frac{1}{2}\begin{bmatrix} \frac{\partial E}{\partial d_x} \\ \\ \frac{\partial E}{\partial d_y} \end{bmatrix} \approx \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-w}^{p_y+w} \left((\nabla I(x,y))^T \mathbf{d} - \delta I(x,y)\right) \nabla I(x,y), \tag{2.12}$$

and we denote

$$G = \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-w}^{p_y+w} (\nabla I(x,y))^T (\nabla I(x,y)), \tag{2.13}$$

$$\mathbf{b} = \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-w}^{p_y+w} \delta I(x,y) \nabla I(x,y), \tag{2.14}$$

thus

$$\frac{1}{2}\frac{\partial E}{\partial d} \approx G\mathbf{d} - \mathbf{b}. \tag{2.15}$$

And since we want the derivative set to zero, we want

$$G\mathbf{d} - \mathbf{b} = \mathbf{0}, \tag{2.16}$$

To get an accurate solution we iterate using Newton-Raphson over $\mathbf{m}^k$, to get the optimal displacement vector $\mathbf{d}$ as per [18], where a new error function $J_k(x,y)$ is defined to minimize

$$\mathbf{m}^k = G^{-1}\mathbf{b}^k, \tag{2.17}$$

where

$$\mathbf{d} = \mathbf{d}_0 + \sum_k \mathbf{m}^k. \tag{2.18}$$

In the context of the pyramidal scheme this displacement vector is calculated in a low-resolution version I_L of the image I, denoted as $\mathbf{d}_L$. This $\mathbf{d}_L$ is then

used as initialization for solving equation (2.18) for I_{L-1}, i.e. $\mathbf{d}_{L-1}$ in the higher resolution image. The optic flow at the point (p_x, p_y) will be contained in $\mathbf{d}_0$ once the full resolution level has been reached.

2.2.2 Other Optical flow algorithms

Alternative optical flow methods include the Horn and Schunk method [14], it is less accurate in localised areas, but is more effective in calculating the global flow. The Lucas-Kanade method was chosen for the background subtraction algorithm for its better accuracy. This method uses sparse optical flow that only focuses on a preset number of visual features (corners and edges) to yield sparse motion fields that are more robust than the dense optical flow that has dense flow fields, that are computed for every pixel in an image, that are more sensitive to appearance variations and local flow discontinuities. The dense optical flow is generally considered best for small image motion (< 10 pixels) and sparse optical flow is more suitable for larger image motion (10s of pixels). Other adaptations must also be considered , for example improving the optical flow by implementing Black and Anadan's robust estimation of optical flow [31] and also the extension to robust estimation by Kim *et al*, [32] whose approach deals simultaneously with motion discontinuities and large illumination variations.

2.3 Mixture of Gaussians (MOG)

When modelling using a single Gaussian probability density function we are unable to model more complex multimodal behaviour. One approach is to use a mixture of Gaussians (MOG). For example in Figure 2.3, a complex annulus data set is efficiently represented by a mixture of 4, 5 and 6 Gaussians respectively, indicated by the ellipses. The fit is determined by the covariances of the individual Gaussians in the MOG.

The technique of MOG applied to background subtraction attempts to model regions of background by a mixture of K Gaussian distributions. Due to increase in computational complexity K is normally restricted to a value between 3 and 5. Different Gaussians represent different aspects of interest of a pixel, such as colour or brightness. A multi-modal approach is used as it is necessary to allow for repetitive movements such as lights flickering and moving foliage.

During training of the MOG it is assumed that for a largely static background (with small movements of trees and changes in weather and lighting effects)

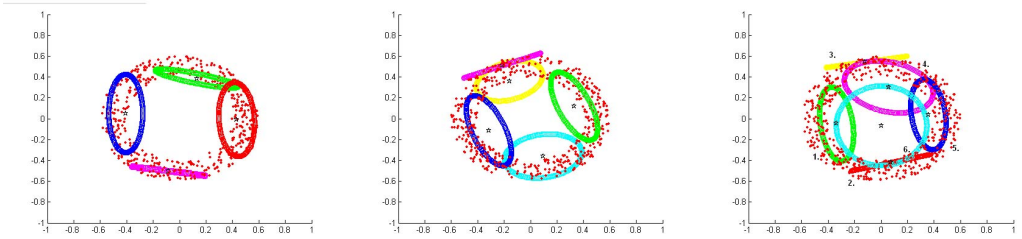


Figure 2.3: Gaussian mixture model fitted to a complex annulus data set using 4, 5 and 6 Gaussians as indicated by the ellipses

the Gaussian components modelling the background will be the ones with the largest weight parameters (longest time in scene). To update the model every pixel in the new image I is checked against the existing model components. If it matches the component it represents it is updated, if not then a new Gaussian component is added to the mixture.

2.3.1 Adaptive Gaussian Mixture Model

Every pixel in an image I is modelled by a mixture of K Gaussian probability density functions. The probability that a pixel has a value of x_t at a time t can be written as

$$P(x_t) = \sum_{i=1}^{K} w_i N_i(x_i; \sigma_i^2 \mathbf{I}, \mu_{i,t}), \tag{2.19}$$

where w_i is the weight parameter of the ith Gaussian component at time t, defined as

$$N_i(x_i; \sigma_i^2 I, \mu_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\sigma_i|} e^{-\frac{1}{2}(x_i - \mu_{i,t})^T (\sigma_i^2 I)^{-1} (x_i - \mu_{i,t})}. \tag{2.20}$$

Here $\mu_{i,t}$ is the mean (center) and $\sigma_i^2 I$ is the covariance (spread) of the ith component. We have assumed a diagonal covariance, sacrificing some accuracy but still providing a satisfactory and fast fit.

In order to simplify the model, the K probability density functions are ranked according to the fitness value $w_{i,t}/\sigma_k$ and only the first N components are used to model the background. The number N is estimated as

$$N = \arg\min_{n}(\sum_{i=1}^{n} w_{i,t} > T), \tag{2.21}$$

where we are trying to estimate the minimum number of components that still accurately models the background. The standard practice is to assume a value for T of 2.5, this means if the MOG modelling the pixel is more than 2.5 standard deviations away from the N distributions, it is marked as foreground. Another important feature of the Gaussian distribution is the fact that, about 68% of values drawn from a standard normal distribution are within 1 standard deviation away from the mean; about 95% of the values are within two standard deviations and about 99.7% lie within 3 standard deviations. This is known as the "68-95-99.7 rule" or the "empirical rule." [33]

The first Gaussian probability density component that matches the test value are updated according to the following equations, where the prior weights of the K distributions are adjusted as follows:

$$w_{i,t} = (1 - \alpha)w_{i,t-1} + \alpha(M_{i,t}). \tag{2.22}$$

$M_{i,t}$ is 1 for the model which matched and 0 for the remaining models. After this approximation the weights are renormalized. The μ and σ parameters for unmatched components remain the same, for a matching distribution the other parameters are updated as follows:

$$\mu_{i,t} = (1 - \alpha)\mu_{i,t-1} + \rho x_t, \tag{2.23}$$

$$\sigma^2_{i,t} = (1 - \alpha)\sigma^2_{i,t-1} + \rho(x_t - \mu_{i,t})^T(x_t - \mu_{i,t}), \tag{2.24}$$

where ρ is defined as

$$\rho = \alpha p(v_{t+1}|\mu_{i,t}, \sigma_{i,t}), \tag{2.25}$$

and $M_{i,t}$ as

$$M_{i,t} = \begin{cases} 1 : \text{if } \omega_i \text{ is the first component to match} \\ 0 : \text{if } \omega_i \text{ does not match} \end{cases} \tag{2.26}$$

where ω_i is the weight of the ith Gaussian component and α is the learning rate, defined as $1/\alpha$, which determines the rate of change of the parameters of

the distributions, in order change the background model to adapt to a changing or a multi-modal background.

If no pixel's distribution matches any of the K distributions, then the least probable component is replaced by a new distribution with the current pixel distribution as its mean and an initially high variance and low weight parameter.

The MOG estimates the mean and variance of each pixel and then tries to reclassify the pixel. These expectation and maximization steps are repeated per pixel until the change in the total GMM likelihood from one iteration to the next is smaller than a certain threshold.

2.3.2 Expectation-Maximisation Algorithm

Expectation step

Given the feature vector, the expectation step calculates the probability of each feature vector being part of each class. This part uses Bayes theorem to calculate the value from the MOG likelihood, weights of each of the probability density functions (pdf) and pdf score of feature vectors when given the class. Each pdf is weighted using the log likelihood equations.

Thus the responsibility p_{kn} of a pdf k to each training data point x_i is:

$$p_{kn} = P(k|x_i) = \frac{p(x_n)P(k)}{p(x_i)}, \tag{2.27}$$

and the GMM likelihood is:

$$p(x_i) = \sum_{k=1}^{K} p(x_i)P(k)p(x_i). \tag{2.28}$$

From equation (2.26) it can be seen that every feature vector influences every pdf. This is the 'soft decision' part of the algorithm. The feature vector is not classified as belonging to a single pdf but belongs to all the pdfs with varying degrees of influence.

Maximization

The Maximization step re-estimates the components of the pdfs that are used in the MOG algorithm, with re-estimation of the mean $\mu_{i,t}$ of PDF k calculated as

$$p(\mu_{i,t}) = \frac{\sum_n p_{kn} x_i}{\sum_n p_{kn}}. \tag{2.29}$$

Normally the mean of a set of data points is

$$p(\mu_{i,t}) = \frac{\int_{-\infty}^{\infty} f(x) x dx}{\int_{-\infty}^{\infty} f(x) dx}. \tag{2.30}$$

Equation (3.32) numerically approximates equation (3.33) using the discrete MOG likelihood values. The likelihood values are the PDF value $f(x)$ in this case.

2.4 AdaBoost

The AdaBoost algorithm [22] solved many of the practical difficulties of earlier boosting algorithms. The algorithm takes as input a training set $(x_1, y_1), ..., (x_m, y_m)$ where every x_i belongs to the same domain X, and each label y_i can take the values of $[-1, 1]$ and repeatedly calls a weak learning algorithm (also called weak learners) in a series of rounds $t = 1, ..., T$, maintaining a distribution of weights $W(i)_t$ over the training set. Initially all the weights are equal, but on progression of the algorithm, the weights of incorrectly classified examples are increased so as to force the weak learning algorithm to focus on the incorrectly classified data in the training set.

The conventional AdaBoost procedure can be easily interpreted as a greedy feature selection process. Consider the general problem of boosting, in which a large set of classification functions are combined using a weighted majority vote. The challenge is to associate a large weight with each good classification function and a smaller weight with poor functions. AdaBoost is an aggressive mechanism for selecting a small set of good classification functions which nevertheless have significant variety. Drawing an analogy between weak classifiers and features, AdaBoost is an effective procedure for searching out a small number of good "features" which nevertheless have significant variety.

In Figure 2.4, the testing data comprises of a cluster of blue points representing class $[-1]$ and another cluster of red data points surrounding the first cluster representing class [1]. For this example a straight line was used as a weak learner, as it gives a 50/50 chance of separating the classes when used on its own, but using AdaBoost with a set of 20 straightline weak learners, it can be

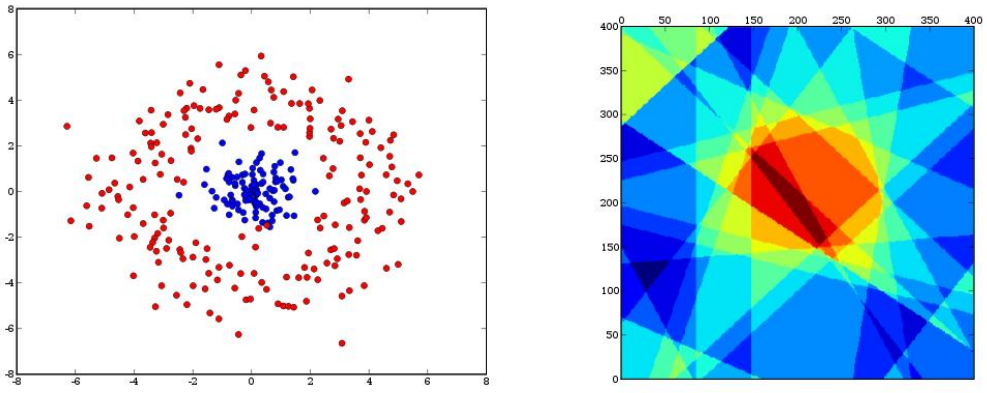


Figure 2.4: Classification of a complex data set using lines as weak learners

seen that the AdaBoost algorithm performs very well in clustering the data points in their respective classes.

The weak learning algorithm must find a weak hypothesis $h_t : X \to [-1, 1]$ suitable for the weight distribution W_t.
How well a weak learning algorithm performs is measured by its error ϵ_t

$$\epsilon_t = Pr_{\text{i in } D_t}[h_t(x_i) \neq y_i] = \sum_{h_t x_i \neq y_i} W_t. \tag{2.31}$$

The weight of this distribution on training example x_i on round t is denoted $W_t(i)$ and is initialised as $W_1(i) = \frac{1}{m}$.

This error ϵ_t is measured with respect to the weight distribution W_t on which the weaklearner was trained.

So thus the AdaBoost algorithm for round $t = 1, ..., T$:

1. Train the weak learning algorithm on the data using the weights W_t

2. Calculate a weak hypothesis $h_t : X \to [-1, 1]$ with error

$$\epsilon_t = Pr_{\text{i in } W_t}[h_t(x_i) \neq y_i] = \sum_{h_t x_i \neq y_i} W_t. \tag{2.32}$$

3. Calculate the learning rate α_t as

$$\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t}). \tag{2.33}$$

4. Update the weights

$$W_{t+1}(i) = \frac{W_t(i)}{Z_t} \times z = \begin{cases} e^{-\alpha_t} : \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} : \text{if } h_t(x_i) \neq y_i \end{cases} \tag{2.34}$$

where Z_t is a normalization factor chosen such that W_{t+1} will be a distribution.

5. The final hypothesis is a weighted combination of all the hypothesis formed by the committee of weak learning algorithms

$$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)). \tag{2.35}$$

Intuitively we can see that when α_t gets larger as the error ϵ_t gets smaller and vice versa, making the algorithm focus more on classifying the foreground (the incorrectly classified) and the background (the correctly classified).

Our foreground object detection procedure classifies foreground objects in images based on the values of simple features contained in the training data set. There are many motivations for using features rather than the pixels directly. The most common reason is that features can encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. For this system there is also a second critical motivation for features: the feature-based system operates much faster than a pixel-by-pixel basis. These features are encoded using the Haar-like wavelets as a combination of simple image features as in [23]. The weak learners used in the algorithm consist of a simple linear set of basic filters, thus for a match a feature must pass through the filters sequence. If any filter in the linear sequence rejects a feature, the feature will not form part of the foreground.

2.5 Summary

In this chapter we presented the most common methods in computer vision for background modeling. We discussed eigenbackgrounds, optical flow, mixture of Gaussians and AdaBoost. We provided an overview of these methods and learnt from their characteristics that they are suitable for our purpose of a robust, accurate and CPU inexpensive algorithm.

Chapter 3

Shadow Removal

3.1 Chromacity measure

Shadow detection and removal is critical to allow background subtraction algorithms perform well and to make subsequent analysis and tracking more accurate. It is desirable to discriminate between foreground objects and their shadows.
Shadows have been modeled as conic volumes [8], this was proven to be effective, but this approach is not computationally feasible for large images as it is a 2 step algorithm and it requires additional calculations per-pixel per step.

Colour information is useful for shadow removal as it separates colour information from lightness information [34].

Given the three colour variables R,G and B, the chromacity coordinates r,g,b and lightness s can be calculated as follows [35]

$$r = \frac{R}{R+G+B} \tag{3.1}$$

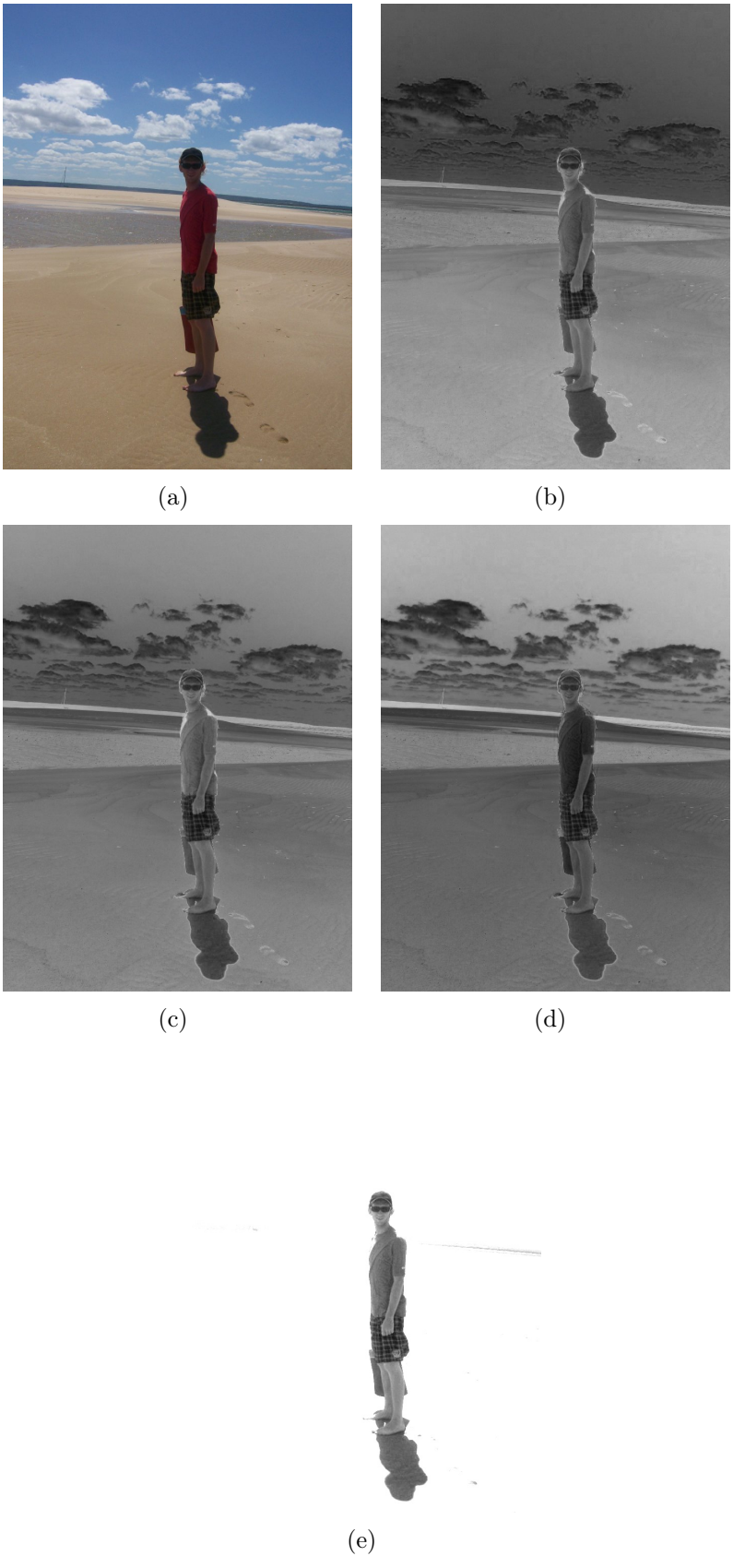
$$g = \frac{G}{R+G+B} \tag{3.2}$$

$$b = \frac{B}{R+G+B} \tag{3.3}$$

$$s = R+G+B \tag{3.4}$$

where $r + g + b = 1$.

When shadows appear or dissappear from the scene, it is usually assumed that the chromacity part at the pixel has not significantly changed. Chromacity coordinates (r, g) has the advantage of being more insensitive to small light changes that are due to shadows, but has the disadvantage of losing lightness information. Thus we adopt the feature space (r, g, s) for test purposes as in [9]. This lightness information is related to the difference in whiteness, blackness and grayness between different objects. Consider the following case where a foreground object (a person) wears a white shirt and walks against a gray background. Since white and gray both have the same chromacity coordinates, the person will not be detected.



(a) (b) (c) (d) (e)

Figure 3.1: Example outdoor scene with a clearly defined shadow and light source broken up into the seperate chromacity coordinates (a), (b) is the R component, (c) is the G component, (d) is the B component and (e) is the Lightness information

3.2 Colour spaces

It is also viable to test the algorithms using different colour spaces, especially the HSV and YCbCr colour spaces. The different colour components are filtered differently to try and eliminate shadow information present in the components.

3.2.1 HSV colour space

The HSV colour space is quite similar to the way in which humans perceive colour. The HSV colour space has three components: hue, saturation and value. This colour space is a transformation of an RGB colour space, and its components are relative to the RGB colour space from which it was derived. Russ [36] suggest the use of a two-dimensional threshold, e.g. a threshold for the hue value and a threshold for the saturation value in an HSV colour space, can have better segmentation success that a single dimensional threshold. On the negative side, the major disadvantage of thresholding is that spatial information is ignored.

(a) (b) (c)

Figure 3.2: Example outdoor scene represented in HSV colour space, where (a) is the H component, (b) is the S component and (c) is the V component.

The HSV colour space of an image is calculated from the RGB colour space as follows

Let r, g, b $\in [0, 1]$ be the red, green, and blue coordinates, respectively, of a color in RGB space.

Let max be the greatest of r, g, and b, and min the least.

To find the hue angle h $\in [0, 360]$ for either HSL or HSV space, compute:

$$h = \begin{cases} 0, \text{if } \max = \min \\ (60^\circ \times \frac{g-b}{\max - \min} + 360^\circ) \bmod 360^\circ, \text{if } \max = \text{r} \\ 60^\circ \times \frac{b-r}{\max - \min} + 120^\circ, \text{if } \max = \text{g} \\ 60^\circ \times \frac{r-g}{\max - \min} + 240^\circ, \text{if } \max = \text{b} \end{cases} \tag{3.5}$$

$$s = \begin{cases} 0, \text{if } \max = 0 \\ \frac{\max - \min}{\max} = 1 - \frac{\min}{\max}, \text{otherwise} \end{cases} \tag{3.6}$$

$$v = \max \tag{3.7}$$

3.2.2 YCbCr colour space

This colour space is not an absolute colour space, it is a way of encoding RGB information. This leads to a practical approximation to color processing and perceptual uniformity, where the RGB colour space is processed into perceptually meaningful information. By doing this, subsequent video processing, transmission and storage can do operations and introduce errors in perceptually meaningful ways. YCbCr is used to separate out a luma signal (Y) that can be stored with high resolution and two chroma components (Cb and Cr) that can be subsampled, compressed, or otherwise treated separately for improved system efficiency.

(a) (b) (c)

Figure 3.3: Example outdoor scene represented in YCbCr colour space, where (a) is the Y component, (b) is the Cb component and (c) is the Cr component.

The YCbCr colour space for an RGB image is calculated as follows

$$Y' = +0.299R + 0.587G + 0.114B \tag{3.8}$$

$$Cb = 128 - 0.168736R - 0.331264G + 0.5B \tag{3.9}$$

$$Cr = 128 + 0.5R - 0.418688G - 0.081312B. \tag{3.10}$$

3.2.3 Other shadow removal methods

There are other ways put forward for shadow removal. In [37], a 1-d illumination invariant shadow-free image is calculated, then used together with the original image to locate shadow edges. By setting these shadow edges to zero in an edge representation of the original image, and by subsequently re-integrating this edge representation by a method resembling lightness recovery, a full colour, shadow free image is created which can then be used in subsequent background subtraction algorithms.

In [38], shadows are removed from a image based on the shadow density, which is defined as a measure of brightness. Using the shadow density, the image is segmented into several regions that have the same density, shadows are then removed by modifying the brightness and color of the regions.

To successfully remove shadows, they must first be identified. Shadow identification may proceed by considering edges of shadows and distinguishing them from all other types of edges. Edges may be classified as either reflectance edges or luminance edges. The difference between a shadow and non-shadow region will only be apparent in luminance, but not in chromatic content and this feature may be exploited for the purpose of shadow detection. The choice of color space in which images are analyzed may make the task of edge classification easier.

Chapter 4

Hardware and Implementation

Two systems were investigated for this project: a distributed system and a web-based system.

4.1 Hardware

4.1.1 Distributed system

The first setup consists of a TK-thick micro computer, Figure 4.1, this is interfaced with a single traditional CCTV camera through a usb framegrabber device, shown in Figure 4.2. In this setup every camera has its own dedicated computer. Messages are then sent to the user's computer alerting them whenever movement was detected in one of the cameras, as can be seen in Figure 4.3. The limitations of this system is the high cost per TK-thick computer and the use of a usb framegrabber device compared to the web-based system discussed next.

4.1.2 Web based system

Although still widely used, CCTV require framegrabbers in order to interface to a personal computer as well as long and expensive cabling. Recently new cameras have become available, such as the Axis 207 and D-Link wireless cameras, that allow for easy setup, lower cost and maintenance. These cameras are very suitable for our application because the cameras run a small embedded webserver interface that streams M-PEG video over TCP/IP, allowing for flexible installation of cameras and uncomplicated computer interfaces. Another option is a cluster of 802.11 wireless cameras around a central wireless router,

Figure 4.1: The TK small footprint computer

Figure 4.2: The USB framegrabber

connected to the server via wired LAN. This option is much more flexible with regards to installation, as only power is needed to allow the cameras to operate, allowing the application to cover a larger area using less cabling. The overview of the web based system can be seen in Figure 4.5.

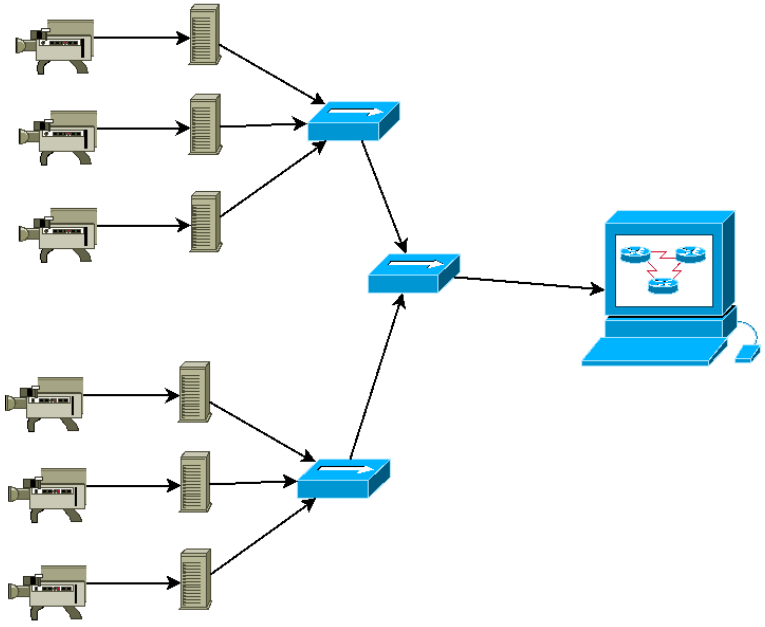


Figure 4.3: Distributed system overview

Figure 4.4: The Axis 207 IP Camera

Of course processing the video streams from so many cameras require a substantial amount of processing power. Recent developments for the commercial market by Intel, such as their *QuadCore* and AMD's $X2$ processors, have made much more computing power available on a smaller scale, allowing for multiple threads to be run on their own processor cores. Even more recently Nvidia has released their *Tesla* computer, a normal desktop sized computer with an amazing 4 Teraflops performance. This performance is achieved using graphics processing units or GPU interfaced by CUDA (Compute Unified Device Architecture) that is a parallel computing architecture developed by

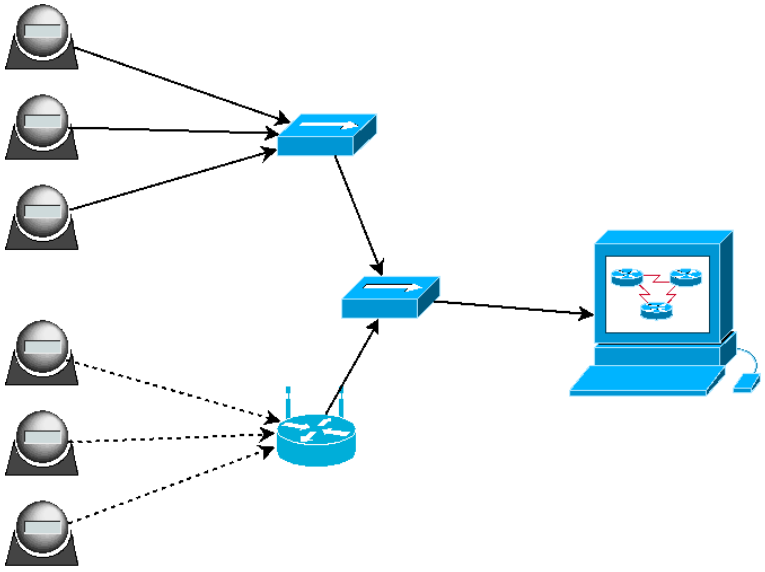


Figure 4.5: Web based system overview

NVIDIA. CUDA is the computing engine in NVIDIA GPUs that is accessible to software developers through industry standard programming languages. The *Tesla* is comparable with a large computing cluster or supercomputer, making it extremely advantageous for realtime computation. One of the obvious benefits for automated surveillance would be that each camera has its own dedicated processor cores. This makes realtime algorithms much more viable than relying on the multiple personnel constantly monitoring the traditional bank of CCTV monitors. [1]

4.2 Software used

4.2.1 NumPy and SciPy

NumPy [39] is a Python library for mathematical and matrix routines that has all of the functionality of commercial software such as Mathematica and Matlab together with SciPy [40] which covers the algorithms and extensions not implemented in NumPy, it also provides GUI construction, plotting graphs and various other scientific routines.

SciKits [41] was also used for its Gaussian and Expectation-Maximisation algorithms.

[1] http://www.nvidia.com/object/teslacomputingsolutions.html

4.2.2 OpenCV

OpenCV [42] is a computer vision library originally developed by Intel. It is free for research and commercial use under a BSD license. The library is cross-platform, and runs on Mac OS X, Windows and Linux. It focuses mainly on real-time image processing, and is used by many computer vision systems such as the vision system of Stanley, the winning entry to the 2005 DARPA Grand Challenge race. OpenCV is widely used in video surveillance systems and is the key tool in the software Swistrack, a tracking tool for understanding self-organization in insects and swarm robotics.

OpenCV also has Python wrappers available for most of its routines. The Intel Performance Primitives (IPP) [43] is a commercial library designed to optimize the running speed of OpenCV by fully utilizing the underlying architecture, this is also available from the Intel website as either a trial version or available for purchase.

4.2.3 Problems encountered

The initial TK-thick problems encountered: The system comes without any operating system installed, after some research we found out that Ubuntu 7.04 (which has local file, python and opencv repositories) supports the low voltage VIA processor and the various other quirks that the system has such as no power management and no ACPI. Ubuntu 7.04 uses minimum system resources and still has ease of installation and a GUI (choices between Gnome and KDE, Gnome is less resource intensive). Linux is a growing environment due to the Open Source philosophy, but the auto software upgrade makes for an unstable distributions because of the differing levels of interaction of various software package versions. This version of Ubuntu was also a Long Term Support (LTS) version, so bugfixes and software upgrades will be implemented for a longer period of time, making for a more stable and workable system in which to develop algorithms.

The OpenCV Python wrappers are also not completed, we had to work around that by writing custom code that allow for Python/C++ interaction. We also had implemented cvMatrix to Numpy matrix conversion as the included *adaptors.py* file from OpenCV is only for Numeric (Numeric was integrated into NumPy 1.0), not the NumPy that is the standard release on Ubuntu 7.04.

The OpenCV recording is also not completed, we had to work around that if we wanted to record or open files with differing CC codes (DIVX, XVID, AVI, MPEG). This was solved by using MPEG encoding as standard for output from the cameras or when using a video file.

4.3 Summary

In this chapter we discussed the available hardware and different system configurations available to us. We also gave reasons why the web-based system was chosen above the distributed system. We also discussed the various software that was used to develop our algorithms and the problems encountered with the hardware and software choices we made.

Chapter 5

Testing and Results

5.1 Overview

To evaluate our background subtraction algorithms, we performed a series of experiments; the results are presented in this chapter.

The primary goal is to see whether the algorithms work on sequences with different noise levels, motion characteristics and appearance of background and foreground components. A second objective is to understand the effects of the parameters of each algorithm. We also test our algorithms using the different colour spaces in Section 5.4 using seperate example sequences in order to try and suppress shadow information. Since these are diverse algorithms, it is difficult to compare them. For a particular application one algorithm might be more suitable than another, for instance in tracking vehicles, optical flow might be more suitable than eigenbackgrounds. Sequences with less complex backgrounds could have been used to test the algorithms, this has been done in other research and we want to see how the algorithms deal with real life problems, i.e. bootstrapping, moved background objects, camouflage, etc.

This section presents a comparative performance analysis based on speed, memory requirements and accuracy, an extensive accuracy analysis is not possible in the scope of this thesis, as it would require agreement on an experimental benchmark or a complex theoretical comparison. Here we limit the discussion to analyse the main algorithm features and shortcomings. The Optical flow algorithm was thresholded on the size of the velocity of each vector so that only relevant foreground information is represented, even then spurious flow elements can be seen.

Our six test sequences are described in Section 5.2, the results are presented in Section 5.3, and indications of computational costs are given in Section 5.6.

5.2 Description of the image sequences

In order to evaluate our algorithms we use a total of six test sequences. They are all 640×480 resolution (standard VGA) AVI files with frame rates at 30 fps. The sequences are all mock-ups of potential surveillance or security setups and settings. Due to hardware limitations, this is the highest resolution the cameras can provide. (For interests sake, the wallpaper test sequences are 180×160 in resolution, from which we can see intuitively that a lot of information has been lost.) Here follows a short description of each sequence with example images.

5.2.1 Test sequence 1

Figure 5.1: Test sequence 1 with “camouflage” problem

An outside sequence of 1657 frames featuring an asphalt parking lot with gardens surrounding it, next to a large building. The parking lot is in the building's shadow. People with dark clothes crossing in and out of the frame; the dark clothing blends in with the dark background making it difficult as the people only "re-appear" in the sunlit areas of the sequence, as seen in Figure 5.1.

5.2.2 Test sequence 2

Figure 5.2: Test sequence 2 with “illumination” problem

A sequence of 9445 frames with the same outdoor setting as sequence 1. As the sequence progresses night falls in the scene with streetlights turning on, a few people move through the scene and some cars move in and out of the frame with headlights throwing moving "pools" of light. This can be seen in Figure 5.2.

5.2.3 Test sequence 3

Figure 5.3: Test sequence 3 with "initialization" problem

A sequence with 795 frames, Figure 5.3, with the same outdoor setting as test sequence 1 and 2. In this sequence the background is partially occluded in the first few frames due to a group of people moving across the scene and parked cars.