Optical vehicle tracking - A framework and tracking solution

November 15, 2004

Ronald Highet
Supervisor: Richard Green
Computer Science Department
Canterbury University
Christchurch, New Zealand

Abstract

There are a large number of different approaches to vehicle tracking currently available. In this report we look at different problems faced when tracking vehicles, such as background subtraction and vehicle recognition, and look at possible solutions to these. We present a layered approach to vehicle tracking, which includes the use of background subtraction based on a statistical colour model, shape approximation using contour creation algorithms, and two dimensional object recognition using colour histograms and geometric moments. We improve on the standard statistical RGB background removal model by adding a second pass HSV shadow removal filter and demonstrate that this provides cleaner background segmentation that other approaches including optical flow. We improve on prior research into simple vehicle tracking solutions by combining object recognition based on both colour histograms and geometric moments, and demonstrate the robust nature of our solution through a number of example scenarios.

Acknowledgements

Firstly I would like to thank my supervisor Richard Green, for many hours of late night proof-reading and enthusiastic advice. Thank you to Oliver Hunt for his informative discussion with me on the general theory of moment functions and help with debugging geometric moment related code. Finally I would like to thank Lara Rennie for proof reading this document.

Contents

List of Figures

Chapter 1

Introduction

"Computer vision as a field is an intellectual frontier. Like any frontier, it is exciting and disorganised; there is often no reliable authority to appeal to– many useful ideas have no theoretical grounding, and some theories are useless in practise; developed areas are widely scattered, and often one looks completely inaccessible from the other"[1]

Computer vision is the process of using statistical models to understand digital images. These models are often constructed with the aid of geometry, physics and learning theory. Computer vision enables us to both make inferences on the real world based on single pixel values and to combine the information obtained from multiple video cameras into a single coherent model. Computer vision also enables us to impose order on groups of pixels to separate them from each other. It enables us to recognise these groups of pixels either based on geometric information such as shape and size or using probabilistic techniques. Computer vision has a large variety of applications including robot navigation, industrial inspection and object tracking. Object tracking adds a temporal aspect to the attributes of computer vision discussed previously; the aim is not simply to assign meaning to objects in a visual scene but also to track their movement through a sequence of images. To achieve this, we must be able to quickly and reliably distinguish between objects in a video steam. An object tracking system[20, 21] must be capable of assigning the same meaning to an object in a digital image independent of its spatial location.

Traffic monitoring systems[13, 36, 33, 38, 34, 19, 8, 18] are software systems designed to collect statistics on traffic in a given geographical region. Traffic types can be classified using the taxonomy illustrated in figure 1.1.

Traffic type	**Description**
Pedestrian	People Walking
Self powered	Bikes, Skateboards, Tricycles
Automotive	Cars, Trucks

Figure 1.1: Traffic taxonomy

This report presents a automotive traffic monitoring framework designed to operate on a standard desktop PC. One primary objective is to keep cost to a minimum and for this reason only a single optical sensor (camera) is used. The field of optically based traffic monitoring systems is well researched and a large number of systems [18, 33, 38, 34, 19, 8, 24, 22] have been presented that tackle this task in a variety of different ways. Our system improves on these by combining a number of different approaches. We initially approached the problem domain by developing a modularised frame work. This modular approach to the development of our tracking system

[1]**Computer Vision - A Modern Approach**[1]

enables us to *'snap together'* different algorithms for each layer without affecting subsequent sections of our system. There are five layers in the system; Video Acquisition, Background removal, Vehicle detection, Vehicle recognition and Vehicle tracking. For each of these layers a number of algorithms have been evaluated and the most promising selected for use in the final system. The final tracking system includes the use of statistical colour-model based-background subtraction *(section 4.2)*, shape approximation using polygonal approximation algorithms *(section 4.4)* and two-dimensional object recognition using geometric moments *(section 4.5)* and colour histograms *(section 4.6)*. The polygonal approximation based approach is similar to that presented in Gerald Kuhne et al. (2001)[9]. However, we significantly improve the reliability by combining it with colour-based object recognition, similar to the methodologies presented in Katja Nummiaro et al (2001)(2003) [15, 16]. We improve on the standard background subtraction algorithm, by adding a second pass shadow removal filter and demonstrate that this provides cleaner background segmentation than other approaches including optical flow.

An introduction to traffic monitoring systems has been presented in section 1. Our report continues in Chapter 2 to introduce a number of previous developed traffic monitoring systems. Chapter 3 presents the motivations and aims of this research. Following this, chapter 4 provides the background information necessary to fully understand the system. Chapter 5 then discusses the development of our traffic monitoring system, providing an overview of the system requirements, the system's design and implementation details. A summary of the research results are then presented in chapter 6. Finally the concluding section of this report summarises the research presented in this report and discusses possible extensions to the systems and future applications of this research.

Chapter 2

Related Work

There is currently a large amount of commercial interest in vehicle tracking and traffic monitoring systems. Traffic monitoring systems are particularly useful in the age of terrorism, to increase the effectiveness of sparsely distributed security forces. The ability to automatically count the number of vehicles that enter or leave an area or that pass a given location, as well as the ability to search for specific types of cars across multiple simultaneous video streams can drastically improve the effectiveness of security personnel. While large numbers of these systems are not publicly released there is still a large amount of research publicly available into the development of similar systems. This chapter discusses a number of these systems that are directly related to our research, and discusses the strengths and weakness.

2.1 View Independent Vehicle/Person Classification[3]

This paper presents an object classification system in digital surveillance that can be used from an arbitrary camera viewpoint. The system is designed to distinguish humans from vehicles for an arbitrary scene. The system employs a two phase approach. In the primary phase, human and vehicle recognition is performed using traditional feature-based classification. This phase initialises the view-normalisation parameters. The second phase performs object classification based on these normalised features. This normalisation also enables absolute identification of size and speed. Size and speed can be used to identify vehicles and search for objects of certain sizes and speeds.

This system has a relatively straight-forward object classification approach. Moving objects are detected by combining evidence from differences in colour and motion. This background subtraction approach is identical to that developed in the paper by Connel, J. et al (2003)[6]. The resulting saliency map is smoothed and holes removed using morphological operators. Several mechanisms are built into this phase of the system that deal with changing lighting conditions and scene composition. Detected objects are tracked using both size and colour properties. Our approach to vehicle tracking is based on the approach presented in this paper, combining both colour and motion cues to produce a robust tracking solution. Our approach does not have to worry about the presence of pedestrians and thus the overall approach is a lot simpler.

2.2 Simultaneous tracking of pedestrians and Vehicles by the Spatio Temporal Markov Random Field Model [33]

This paper presents a traffic monitoring system capable of tracking both pedestrians and vehicles similar to the system presented by Lisa M. Brown (2003)[3]. The system uses a predefined Spatio-Temporal Markov Random Field Model *S-T MRF* to divide an image into blocks of related

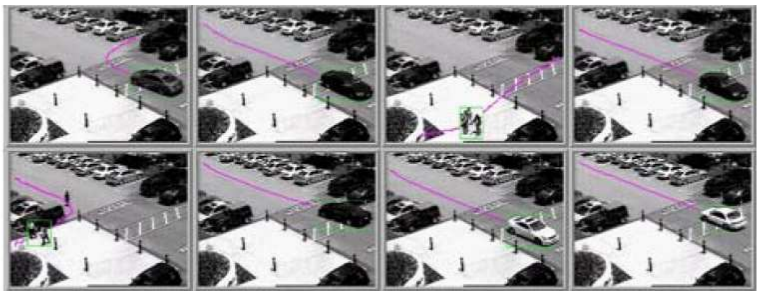

Figure 2.1: Sample trackign results from view independant tracking system

pixels. The system performs background subtraction by only processing those blocks which have a significantly different texture to that of the background.

The system defines the following functions:

1. $G(t-1) = g, G(t) = h$; An image G at time $t-1$ and t is defined as having a value g and h respectively. The values of each pixel can be defined by the equation $G(t-1; i, j) = g(i, j), G(t; i, j) = h(i, j)$
2. $X(t-1) = x$, $X(t) = y$; An object map X at time $t-1$ and t is estimated to have a distribution of labels x and y respectively. For each block in the map this condition can be described using the following equations: $X_k(t-1) = x_k, X_k = y_k$ where k represents the block number
3. $V(t-1); t = v$ A motion-vector map V from time $t-1$ to t is estimated with respect to each block.

Using the **S-T MRF** model, the system simultaneously determines the Object map $X(t)$ and the Motion vector field $V(t)$, which are then used to calculated a Maximum A posteriori Probability field. The paper improves on the stanard **S-T MRF** by simultaneously optimising segmentation boundaries and motion vectors by both referring to texture and labelling correlations across a video stream (across the temporal axis). The results show that the system is very robust for tracking flexible and fixed objects such as vehicles and pedestrians, even with frequent occlusion.

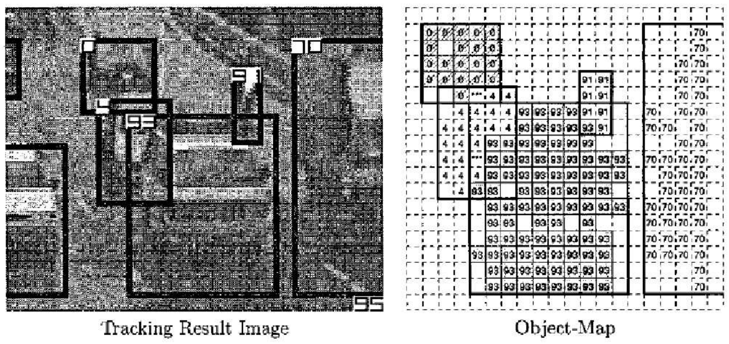

Figure 2.2: Same results from spatio temporal random field model based tracking

Chapter 3

Research Goals

This chapter introduces research into vehicle tracking. Section 3.1 introduces the motivations behind this research project, discussing the reasons why this project was undertaken. Section 3.2 then presents the objectives or aims of this research project.

3.1 Motivation

The motivations behind this research project are two-fold; There is firstly much commercial interest in tracking solutions. Simultaneously there is a the lack of readily available simple approaches to vehicle tracking. These motivations are further discussed below.

3.1.1 Commercial Interest

Since September 11 2001, there has been a large interest in terrorism prevention. The United States Department of Defense has allocated many millions of dollars to the development of systems that are capable of tracking vehicles and people. These systems have been deployed throughout The United States, both in airports and other public facilities. While these have largely been operational failures, these projects have served as a catalyst for commercial interest into security systems based on computer vision. In Christchurch New Zealand, a number of companies are looking at the development of systems capable of tracking vehicles across a number of video sources. The aim is to develop a system that would lighten the load on security forces, by automating as much of the surveillance work as possible. The system developed in this paper prototypes a number of the facets of such a security system and provides a base for future development.

3.1.2 Lack of simple solutions

Most vehicle tracking solutions presented in research publications are highly specialised and the algorithms developed in these papers are never extended. Large numbers of these algorithms achieve the same results, yet there is no comparison between the algorithm presented and similar past research. The motivation behind this project is to evaluate a large number of possible solutions to problems encountered when building a optical tracking system, and to select the most promising solution for further development. This avoids the unnecessary development of solutions to problems which have already been solved.

3.2 Objectives

The objectives of this research project are as follows:

1. Develop a framework for the development of vehicle tracking applications. This framework should be platform independent and support the rapid prototyping of vehicle tracking solutions.

2. Using the aforementioned development framework, develop a simple vehicle tracking system that uses a single camera and relies on optical cues for tracking.

3. Utilize the development of the tracking system to evaluate a number of recent approaches to vehicle tracking, and select the most promising approaches for further development.

Chapter 4

Background

This chapter introduces relevant background information. These are concepts that are critical to the development of the tracking solution. OpenCV, a computer vision software development kit, used as a base for system development, is presented in section 4.1. A simple approach to background subtraction is presented in section 4.2. Another technique used in system development for both background subtraction and object identification, called optical flow is presented in section 4.3. Polygonal approximation algorithms are covered in section 4.4, followed by a discussion of moment functions in section 4.5. Finally, colour histograms and colour-based backprojection of images are discussed in section 4.6.

4.1 The OpenCV library

The OpenCV library is a software library designed to aid in the development of software projects involving computer vision.The OpenCV library implements a wide variety of tools for computer vision applications. It is compatible with the Intel Image Processing Library *(IPL)*, which implements a large range of very low level operations for manipulation of digital images. OpenCV provides a small range of primitive functions, but is in essence a high-level library. Table 4.1 lists a number of functions provided by the OpenCV library.

OpenCV Features
Camera calibration
Optical Feature detection
Optical Feature tracking including Optical Flow
Shape Analysis including Geometry and Contour Processing functionality
Motion Analysis including Motion templates and estimators
Three-dimensional reconstruction including View morphing
Object Segmentation
Object recognition including Histograms
Markov models and Eigen objects

Figure 4.1: Features provided by the OpenCV library

4.1.1 Data Types

There are a few fundamental and helper data types supplied with the OpenCV library. A number of the fundamental data types provided by OpenCV, these are summarised in figure 4.2.

The presence of these data types simplifies the development of computer vision projects as it minimizes the amount of low level coding. This is especially useful in projects similiar in nature

Data Type	**Description**
IplImage	Generic image structure
CvMat	Generic matrix structure
CvSeq	Deque collection
CvSet	Set structure
CvGraph	Generic graph structure
CvHistogram	Generic histogram structure
CvPoint	Two-dimensional point
CvMoments	Spatial moment structure

Figure 4.2: Data types provided by OpenCV

to ours, because more time can be devoted to research and implementation of new algorithms.

4.1.2 Error Handling

OpenCV uses a set of global error status flags that can be set or retrieved using presupplied functions **cvError** or **cvGetErrStatus**. The error-handling mechanisms are fully adjustable and user-specified.

4.1.3 Hardware Requirements

The OpenCV library is capable of running on most personal computers. The OpenCV library is only optimized for use of Intel-based systems and as such it is highly recommened that an Intel System be used for any OpenCV software development. The OpenCV library is available on both Windows and Linux platforms. The OpenCV library is supplied as a package with the Debian distribution of Linux.

4.1.4 Software Requirements

The OpenCV library is written in C++ and as such could feasibly be ported to any system that has a C++ compiler. The OpenCV library is supplied with a Visual Studios 6 Project file. However there are a number of the functions used throughout the library that are not compatible with Microsoft's latest Visual Studio .Net. This problem was remedied early in the development of the system with each of the problem functions rewritten so that they work within the .Net environment.

4.2 Background Subtraction

Background subtraction, also known as *background segmentation* and *motion segmentation*, is the removal of sections of the source video that are not relevant to the project. In the case of our tracking system, the term *'background'* is defined as the set of motionless pixels, or pixels that do not belong to any object moving laterally across the field of view of our camera. In many situations, background subtraction can be achieved by subtracting an estimate of the appearance of the background from the image and looking for large absolute values in the result. The main issue associated with these types of background subtraction is obtaining a good estimate of the background. Different background subtraction methods all provide different solutions to this problem. A number of different approaches to background subtraction are presented below.

4.2.1 Simple Background subtraction

The simplest background model assumes that the brightness of every background pixel varies independently, according to a normal distribution. Background characteristics can be calculated by accumulating a large number of video frames, as well as their squares. This means that is necessary to find the sum of pixel values in the location $scene_{x,y}$ and a sum of square of the values $scene^2_{(x,y)}$ for every pixel location. Given that N represents the number of video frames collected so far, the mean pixel intensity can be calculated using the following equation:

$$mean(x,y) = \frac{\sum scene(x,y)}{N}$$

Standard deviation of pixel intensity at that point can be calculated as follows:

$$\sigma(x,y) = \sqrt{\left(\frac{\sum scene^2_{(x,y)}}{N} - \left(\frac{\sum scene_{(x,y)}}{N}\right)^2\right)}$$

After this, a pixel is regarded as being in motion if it satisfies the following condition:

$$\mid \left(mean_{(x,y)} - scene_{(x,y)}\right) \mid > 3\sigma_{(x,y)}$$

This approach produces poor results however, as the background typically changes slowly over time. Examples of these changes would be a road as it rains and less as the weather dries up. The algorithm can be improved by using a *moving average*. In this approach, we estimate the value of a particular background pixel as a weighted average of the previous values. Typically pixels in the distant past should be weighted with 0, and the weights should smoothly increase. The moving average should track the changes in the background, meaning that if the weather changes quickly, relatively few pixels should have nonzero weights, and if changes are slow then the number of nonzero weighted pixels should increase. The background subtraction formulas based on a running average is shown below:

$$mean_{(x,y)} = \frac{1}{N}\sum_{j=i}^{i+N-1} scene_{(x,y)}$$

4.3 Optical Flow

Optical flow[11] is the apparent motion of brightness patterns in an image. Optical Flow algorithms produce a velocity vector field (motion field), that can warp one image to another. An example of two images and the corresponding vector field is shown in figure 4.3

These vector fields can be used for object detection and tracking by grouping vectors of similar length and direction. These vector fields can also be used for background subtraction, by regarding those pixels with a small associated vector as being part of the background. The seminal paper on optical flow by Horn and Schunck (1980)[12] provides a reliable algorithm for calculating optical flow motion vector fields. The process of calculating optical flow is now described.

Assuming that the intensity of an image can be represented as a two-dimensional density distribution $I(x,y)$, which varies over time t, we can define the function $I(x,y,t)$, where the intensity is a function of t, as well as of x and y. Further assumptions made are detailed below.

- Brightness $I_{(x,y,t)}$ varies smoothly according to the values of the x and y coordinates across the majority of the image

- Brightness of every point of a moving or static object does not change over time.

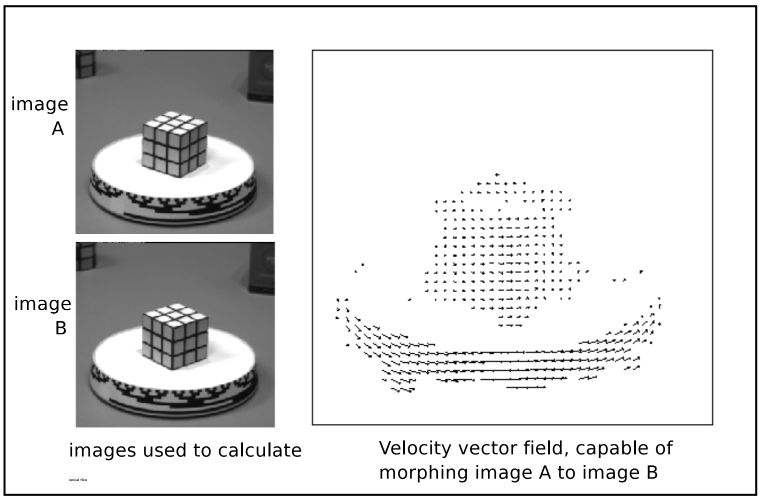
image A
image B
images used to calculate
optical flow
Velocity vector field, capable of morphing image A to image B

Figure 4.3: Optical flow example

To calculate the change in intensity over time, we differentiate with respect to time.

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \tag{4.1}$$

$$\frac{dI}{dt} = \frac{\delta I}{\delta x}\frac{dx}{dt} + \frac{\delta I}{\delta y}\frac{dy}{dt} + \frac{\delta I}{\delta t} \tag{4.2}$$

We now assume that the image intensity of each visible scene point does not change over time *i.e that shadows and illumination are not changing due to motion.* This leads to the following equation:

$$\frac{dI}{dt} = 0 \tag{4.3}$$

This implies that

$$0 = \frac{\delta I}{\delta x}\frac{dx}{dt} + \frac{\delta I}{\delta y}\frac{dy}{dt} + \frac{\delta I}{\delta t} \tag{4.4}$$

After dividing by dt, we have

$$\frac{dx}{dt} = u \tag{4.5}$$

$$\frac{dy}{dt} = v \tag{4.6}$$

u and v represent the speed at which the pixel is moving in the x and y directions respectively. Thus, because the limit of dt tends towards zero

$$I_x u + I_y v + I_t = 0 \tag{4.7}$$

where the partial derivatives of I are denoted by the subscripts $I_{x,y,t}$, and u and v are the components of the optical flow motion vector. This equation (Equation 4.3) is known as the **Optical flow constraint equation** as it expresses a constraint on the components u and v of the optical flow vector field. The optical flow constraint equation is often rewritten as shown below:

$$(I_x, I_y) \times (u, v) = -I_t \tag{4.8}$$

or

$$-\frac{\delta I}{\delta t} = \frac{\delta I}{\delta x}u + \frac{\delta I}{\delta y}v \tag{4.9}$$

$\frac{\delta I}{\delta t}$ of a given pixel represents how fast the intensity is changing over time and $\frac{\delta I}{\delta x}$ and $\frac{\delta I}{\delta y}$ represent how rapidly intensity changes along the x and y axes respectively. We wish to calculate u and v. However, we only have one equation per pixel and two unknown variables, so we cannot unambiguously calculate out vector components. An example of this is shown in Figure 4.4.

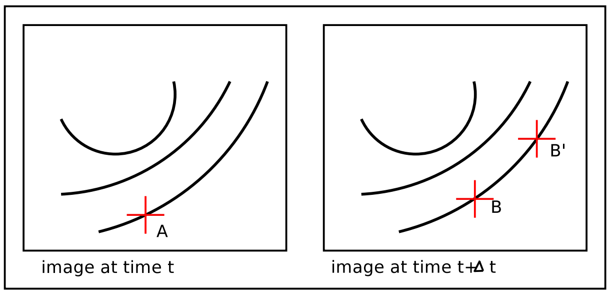

Figure 4.4: Ambiguity in determining optical flow

The lines are contours with identical intensity in both images. The ambiguity occurs when attempting to tell whether the part of the image represented by point A in the first image has moved to point B, B' or any other point of the same intensity in the second image. We require additional information to unambiguously determine the values of u and v for each vector in our optical flow vector field. A partial solution to this problem exists in the observation that the motion that is observed in adjacent pixels is very similar, except near the edges of moving objects. A measure of how much the optical flow deviates from the smoothly varying ideal can be calculated by evaluating the following integral over the whole image:

$$S = \int\int_{\textbf{image}} (\frac{\delta u}{\delta x})^2 + (\frac{\delta u}{\delta y}^2) + (\frac{\delta u}{\delta x})^2 + (\frac{\delta v}{\delta x}^2) \tag{4.10}$$

The smoothness constraint represented in equation 4.3 sometimes conflicts with the optical flow constraint. It is possible to show how the solution provided by our smoothness constraint deviates from the condition required by the optical flow constraint by evaluating the following equation:

$$c = \int\int_{\textbf{image}} \left(\frac{\delta I}{\delta x}u + \frac{\delta I}{\delta y}v + \frac{\delta I}{\delta t}\right)^2 dx\,dy \tag{4.11}$$

In order to combine these two constraint equations, it is necessary to use a technique called **Largrangian multipliers[31]**. We attempt to find a solution for u and v which minimises $S+kC$, where k is a scalar multiplier. Typically k is large if our intensity measurements are accurate *(when C is small)* and small if our accuracy is low *(when C is large)*. Minimising the resulting integral produces the following functions for u and v which need to be satisfied:

$$\frac{\delta^2 u}{\delta y^2} + \frac{\delta^2 u}{\delta y^2} \quad = \quad k(\frac{\delta I}{\delta x}u + \frac{\delta I}{\delta y}v + \frac{\delta I}{\delta t})\frac{\delta I}{\delta x} \tag{4.12}$$

$$\frac{\delta^2 v}{\delta y^2} + \frac{\delta^2 v}{\delta y^2} \quad = \quad k(\frac{\delta I}{\delta x}u + \frac{\delta I}{\delta y}v + \frac{\delta I}{\delta t})\frac{\delta I}{\delta y} \tag{4.13}$$

The optical flow motion vector field can then be produced by iteratively stepping across the image and solving the equations for each pixel, using the intensity$I$ derived for each pixel obtained from the original image and the weighting factor k produced previously.

4.4 Polygonal Approximation

Polygonal approximation algorithms aim to approximate the shape of a curve using a number of fundamental mathematical functions. They provide simple representations for relatively complex objects, and produce data that can easily be processed further. The fundamental idea behind all of these algorithms is to find and keep only the dominant points of any curve. These are those points which are critical in defining the shape of the curve, and are hence points where the local maximums of curvature angle are located on the digital curve. In order to approximate a curve, it is necessary to first represent the curve discretely. The curvature of a continuous curve at any one point can be determined as the speed at which the tangent angle is changing, as shown below:

$$k = \frac{x\text{'}y^n - x^n y\text{'}^{\frac{3}{2}}}{(x\text{'}^2 + y\text{'}^2} \tag{4.14}$$

This continuous curve can be quantised using a simple approximation called **L1 curvature**:

$$c_i = ((f_i f_{i-1} + 4)mod 8) - 4 \tag{4.15}$$

This method produces a value varying between 0 and 4 which represents the change in curvature between two points on the curve. 0 represents no change in curvature *(a straight line)* and 4 represents a complete reversal in direction. A number of more complex algorithms can be used to get more accurate discrete approximations of curves. A selection of these are discussed below.

4.4.1 K-Cosine Algorithm

For any given point on the curve (x_i, y_i) the radius m_i represents the neighbourhood of points that can be selected as the next node in the curve. Some algorithms use a constant value of m_i for all points on the curve, while many other algorithms calculate a suitable value of m_i for every point on the curve. The following values are calculated for all pairs (x_{i-k}, y_{i-k}) and (x_{i+k}, y_{i-k}) [1]:

$$c_{ik} = \mathbf{cos}(a_{ik}, b_{ik}) \tag{4.16}$$

where

$$a_{ik} \quad = \quad (x_{i-k} - x_i, y_{i-k} - y_i) \tag{4.17}$$
$$b_{ik} \quad = \quad (x_{i+k} - x_i, y_{i-k} - y_i) \tag{4.18}$$

The index value h_i is then calculated such that the constraint $c_{im} < c_{im} - 1 < ... < c_{ih_1} \geq c_{ih_1 - 1}$. The value c_{ih_1} is regarded as the curvature value of the i^{th} point on the curve. The value of c_{ik} can vary between -1 and 1, representing minimum curvature and maximum curvature respectively.

[1]k= 1...m

4.4.2 Rosenfeld-Johnston algorithm[27]

This algorithm is one of the earliest algorithms for polygonal approximation. The algorithm requires that the parameter m be between $1/10_{th}$ and $1/15_{th}$ of the number of points on the digital curve. The algorithm removes all points from the curve that do not satisfy the following constraint:

$$\ni j, \mid i - j \mid \leq \frac{h_i}{2} \tag{4.19}$$

where $c_{ih_i} < c_{jh_1}$. The remaining points are treated as the dominant points on the curve.

The primary weakness of this algorithm is the necessity to choose the parameter m and the parameter identity must be chosen for all points, which can result in either very rough approximations or excessively precise approximations.

4.4.3 Teh Chin Algorithm[5]

This polygonal approximation algorithm was proposed by C.H. Teh (1989)[5]. The algorithm makes several passes through the curve and deletes some points at each pass. At first, all points with zero curvature are removed. For all other points the parameter M_1 is calculated along with the angle of curvature C_i. After that the algorithm performs a non-maxima suppression, deleting points whose curvature satisfies the previous condition where, for C_i^1, the metric H_1 is set to M_1. Finally, the algorithm replaces groups of two successive remaining points with a single point and groups of three or more successive points with a pair of the first and the last points. This algorithm does not require any input apart from the original curve representing the outline of the object we are wishing to represent.

4.5 Moment Functions

Moment functions are used throughout image analysis. Examples of applications are object classification, object recognition, pose estimation, image compression and invariant pattern recognition. A set of moments calculated for a given image generally represent global characteristics of the image shape and reveal a large amount of information about geometric features of the image. This feature representation capability has been used in computer vision and robotics for object recognition. This project utilises the computationally inexpensive Geometric moments (4.5.1) for object recognition.

An image can be represented as a two-dimensional density distribution $f(x, y)$, where $f(x, y)$ represents the intensity at the given pixel co-ordinate (x, y). If ζ represents the domain of the function $f(x, y)$ (the size of the image in pixels), a general definition of all moment functions ϕ_{pq} of order $(p + q)$ is shown below:

$$\phi_{pq} = \int\int_{\zeta} \psi_{pq}(x, y) f(x, y)\, dx\, dy \qquad p, q = 0, 1, 2... \tag{4.20}$$

where $\psi_{pq}(x, y)$ is the *moment weighting kernel*, a continuous function of (x, y). The indices (p, q) of this function denote the degrees of the coordinates (x, y), as defined inside the function ψ.

The first and most significant paper regarding geometric moments was published by Hu [14] in 1962. In this paper Hu used the geometric moments to generate a set of invariants that could be used for automatic character recognition. The geometric moments have also been used in a number of different object recognition applications, such as ships by Smith [7] and recognising aircraft by Dydabu [29] in 1977.

4.5.1 Geometric moments

Geometric moments are the simplest of the moment functions. Their kernel is defined as the product of the pixel coordinates. The primary advantage of these geometric moments is that transforms in the image space can be easily expressed in terms of corresponding transforms in the moment space. Geometric moments are computationally less expensive than other moment calculations due to the simplicity of their kernel function. Functions of geometric moments that are invariant in the image plane are used throughout computer vision for object recognition.

Geometric moments are defined with the base set $x^p y^q$. With the $(p+q)^{th}$ order 2D geometric moment denoted as m_{pq}, they can be expressed as follows:

$$m_{pq} = \int \int_{\zeta} x^p y^q \, f(x,y) \, dx \, dy \qquad p,q = 0,1,2,3... \tag{4.21}$$

where ϕ represents the image space in which the intensity function $f(x,y)$ is defined.

Each order of geometric moments represents different spatial characteristics of the image for which they were calculated. A set of these moments of different orders can therefore be used to create an overall descriptor of the image's shape. A number of the early order geometric moments are explained below.

The order zero geometric moment m_{00} represents the total intensity of the image. For a silhouette image, m_{00} represents the total area of the image region.

The first order geometric moments m_{01} m_{10}, represent the intensity moment along the x and y axises respectively. The center of mass or *centroid* (x_0, y_0) of an image can be calculated from m_{00},m_{01} and m_{10} as follows:

$$x_0 = \frac{m_{10}}{m_{00}} \qquad y_0 = \frac{m01}{m_{00}} \tag{4.22}$$

$$\textbf{centroid} = (x_0, y_0) \tag{4.23}$$

The second order moments measure the variance of the image's intensity distribution around the origin of the image.

Central geometric moments

Given that (x_0, y_0) represents the optical center of the image, we can shift the origin of our reference system to the optical center of our image. This transformation means that the moment computation will be independent of the images reference system. Moments computed with respect to the optical centroid are called *central moments*, defined as follows:

$$\eta_{pq} = \int \int_{\zeta} (x - x_0)^p * (y - y_0)^q f(x,y) \, dx \, dy, \qquad p.q = 0,1,2,3... \tag{4.24}$$

The order zero central moment η_{00} is therefore directly equivalent to m_{00}. The first order central moments η_{10} and $\eta 01$ are both equal to zero. The second order central moments η_{20} and η_{02} measure the variance in the intensity distribution around the optical center of the image, and covariance measure is given by η_{11}.

Functions of these geometric moments, which are invariant with regard to image-plane transforms, are used throughout object identification. The central moments are by definition invariant to translation. This is because the image's centroid moves with the image during translation, and central moments are defined with the centroid as their origin. It is possible to calculate the second and third order central moments from the ordinary geometric moments as shown below:

$$\begin{aligned}
\eta_{02} &= m_{02} - y_0 m_{01} && (4.25)\\
\eta_{20} &= m_{20} - x_0 m_{10} && (4.26)\\
\eta_{11} &= m_{11} - y_0 m_{10} && (4.27)\\
\eta_{30} &= m_{30} - 3x_0 m_{20} + 2x_0^2 m_{10} && (4.28)\\
\eta_{03} &= m_{03} - 3y_0 m_{02} + 2y_0^2 m_{01} && (4.29)\\
\eta_{21} &= m_{21} - 2x_0 m_{11} - y_0 m_{20} + 2x_0^2 m_{01} && (4.30)\\
\eta_{12} &= m_{12} - 2y_0 m_{11} - x_0 m_{02} + 2y_0^2 m_{10} && (4.31)
\end{aligned}$$

M. Hu (1962) [14] has shown that a set of seven features derived from the second and third-order centralised geometric moments is invariant to translation, rotation, and scale change. These seven features are calculated as follows:

$$\begin{aligned}
h_1 &= \eta_{20} + \eta_{02} && (4.32)\\
h_2 &= (\eta - \eta_{02})^2 + 4\eta_{11}^2 && (4.33)\\
h_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 && (4.34)\\
h_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 && (4.35)\\
h_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 && (4.36)\\
&\quad -3(\eta 21 + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} && (4.37)\\
&\quad +\eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (4.38)\\
h_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21}] + \eta_{03})^2] && (4.39)\\
&\quad +4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) && (4.40)\\
h_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}[(\eta_{30} + \eta_{12})^2 && (4.41)\\
&\quad -3(\eta_{21} + \eta_{03}^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[&& (4.42)\\
&\quad 3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (4.43)
\end{aligned}$$

From these seven invariant feature descriptors three similarity measures I_1,I_2 and I_3 can be calculated as shown below:

$$\begin{aligned}
I_1(A,B) &= \sum_{i=1}^{7} \mid \frac{-1}{m_i^A} + \frac{1}{B_i^B} \mid && (4.44)\\
I_2(A,B) &= \sum_{i=1}^{7} \mid -m_i^A + m_i^B \mid && (4.45)\\
I_3(A,B) &= \mathbf{max} \mid \frac{(m_i^A - m_i^B)}{m_i^A} \mid && (4.46)
\end{aligned}$$

where [2]

$$\begin{aligned}
m_i^A &= \mathbf{sgn}(h_i^A)\mathbf{log}_{10} \mid h_i^A \mid && (4.47)\\
m_i^B &= \mathbf{sgn}(h_i^B)\mathbf{log}_{10} \mid h_i^B \mid && (4.48)
\end{aligned}$$

4.6 Colour Histograms and Back projection

A histogram is a discrete approximation of a continuous variable probability distribution. Histograms are widely used in image processing and computer vision. Examples of uses of one

[2]sgn() determines the sign of a number. Returns 1 if number is positive; 0 if number is 0; -1 if number is negative.

dimensional histograms include: Grey scale image enhancement, Automatic threshold calculation, Selecting colour histograms via hue histograms back projection. Multi dimensional histograms are also used in a variety of situations including: Analyzing and segmenting color images, normalized to brightness (e.g. red-green or hue-saturation images), Analyzing and segmenting motion fields (x-y or magnitude-angle histograms), Analyzing shapes,Content based retrieval, Bayesian-based object recognition

Histograms represent a simple statistical description of an object, e.g An image. The objects characteristics are measured by iterating through that object. For example colour histograms for an image are built from pixel values in one of the colour spaces. All possible values of that multi-dimensional characteristic are further quantized on each coordinate. The histogram can be viewed as a multi-dimensional array. Each dimension of the histogram corresponds to a different object feature for example the three colour spaces of an object in HSV colour format. Each array element represents a histogram bin and contains the number of measurements done of the object with a quantised value equal the to bins identifier. Histograms can be used to compare respective object as follows:

$$D_{L_1}(H,K) = \sum_i \mid h_i - k_i \mid \tag{4.49}$$

$$\tag{4.50}$$

or

$$D(H,K) = \sqrt{(h-k)^T A (h-k)} \tag{4.51}$$

$$\tag{4.52}$$

These methods suffer from a number of severe problems. The metric D_L sometimes gives too small difference when there is no exact correspondence between histogram bins, that is, if the bins of one histogram are slightly shifted. D_L can also indicate a significant difference inaccurately due to cumulative error properties. Rubner Jan et al[39] presents a robust metric which significantly reduces the errors associated with matching histograms, called the Earth Mover Distance.

4.6.1 Colour based Back projection

Back projection is a technique whereby a image can be selectively masked according to a given colour distribution. Given an image in the HSV colour format, and a one dimensional histogram representing the Hue distribution of the colour mask it is possible to black out all pixels which have a Hue value that does not occur above a threshold in the mask histogram. This has the effect of leaving only those pixels visible which have similar colour to the distribution described in the histogram.

Chapter 5

System Development

The primary purpose of this project is to develop a simple yet robust vehicle tracking system. The basic design of this system is discussed in section 5.2. Section 5.4 discusses the implementation in much greater detail, looking at each of the four major layers in the system individually. Section 5.4.1 looks at the video acquisition system. The background removal sub-system is discussed in section 5.4.2. Background subtraction is essentially the removal of sections of the source video that remain constant across frames and are therefore deduced to be non-moving. Section 5.4.3 then discusses our approach to vehicle detection. Vehicle detection is the detection of vehicles in the source video (after background subtraction). Vehicle recognition is discussed in section 5.4.4.

5.1 Requirements

The following section covers the three primary requirements of our system. These requirements are that the system be inexpensive, simple and compatible with the OpenCv software development kit.

5.1.1 Low Cost

A large proportion of vehicle tracking and traffic monitoring systems[24, 38, 30] rely on custom-built hardware and elevated platforms. The cost associated with such projects can often run into hundreds of thousands of dollars[24]. We aim to produce a system capable of being placed beside any two-lane road, with little or no setup cost involved. To maintain this low overall cost, only off-the-shelf, components have been used in the development of this project. Expenditure has been kept to a minimum by using a standard laptop computer and a single digital video camera. Section 5.3 covers the exact specifications of the required hardware in more detail.

In addition to monetary cost, a significant amount of effort has been applied to avoid unnecessary software development time. The OpenCV library discussed previously in section 4.1 was selected as a base for system development.

5.1.2 Computational Efficiency

The system should be kept as simple as possible without affecting the performance or quality of the tracking system. This means that we favour algorithms that require little or no manual calibration. Many similar tracking systems are capable of producing impressive results, but require extensive manual set up. This setup often includes a of choosing thresholds based on “trial and error”. This affects not only the initial deployment cost of any such system, but if these thresholds are incorrectly set it significantly affects the performance of the system.

5.1.3 Sofware Compatibility

The tracking framework for efficient implementation is designed to be compatible with the OpenCv software. This means it must be developed exclusively in C++ and be compatible with both Windows and Linux versions of the OpenCv library.

5.1.4 Modularity

It is an important priority that the framework be distinctly segmented. Each segment or 'module', should be capable of operating independently of those around it. Each module should have a predefined input data format and a predefined output data format shown in figure 5.2. This modular approach to the development of our tracking system enables us to 'snap together' different algorithms for each layer without affecting subsequent sections of our system.

5.2 System Design

This section discusses the design of our tracking framework and looks at the aforementioned system requirements that have been realised in the software solution. The main driving force behind the design of the framework is that it should simplify further system development. The requirement that the system be modular is pivotal to support easy development. By having a modular design we minimise the amount of effort required to test new solutions for each module. There are five primary modules in the system; Video Acquisition, Motion Segmentation, Vehicle detection, Vehicle recognition and Vehicle tracking.

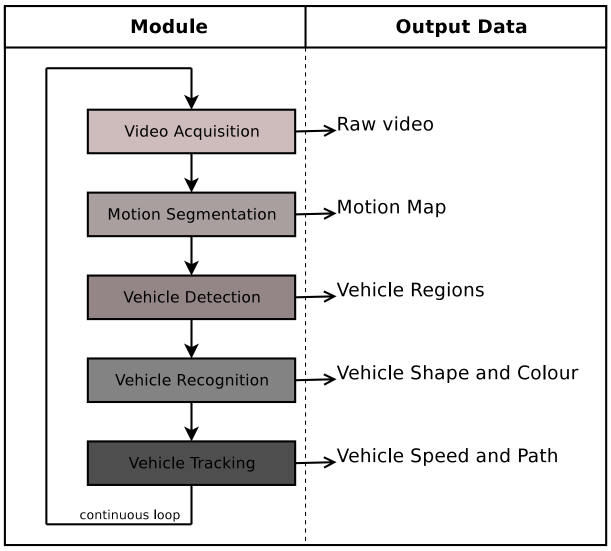

Figure 5.1: Overview of System Modules

Each module can be developed completely separately from all other modules in the tracking system and modules can be easily interchanged.

Each of these modules has distinct responsibilities which are discussed in the following section:

5.2.1 Video Acquisition

This module is responsible for retrieving each of the video frames from the camera. The input data format for this module can vary as a number of video cameras such as the ADS USB 2.0 or the Sony Handcam produce full resolution uncompressed video, while a number of other cameras

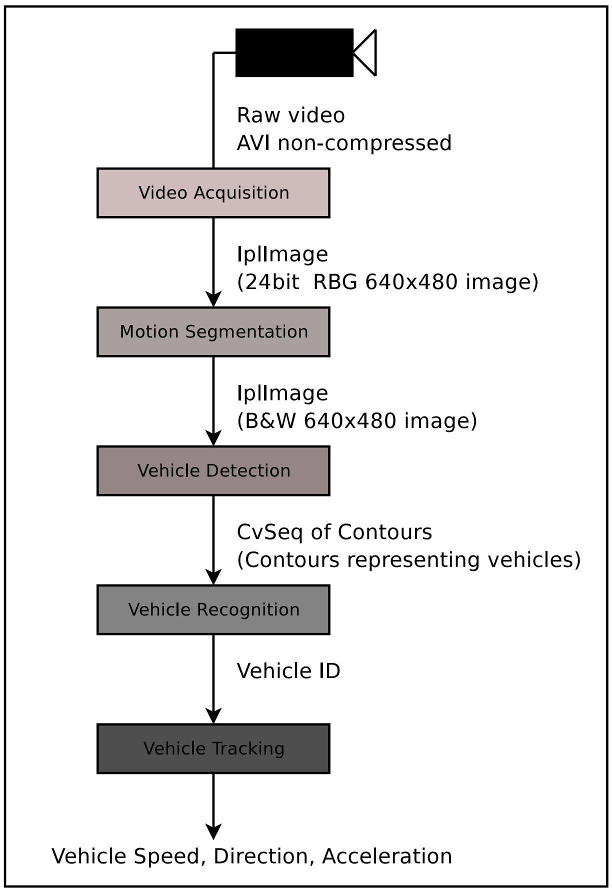

Figure 5.2: Output data format

including the Sony Eye Toy and Logitech Web Cam produce compressed video. The output format is a stream of **IplImage**s, which are the generic image format supported by OpenCV. This means that modules handling interaction with compression-enabled cameras must uncompress the video before producing the image stream.

5.2.2 Motion Segmentation

This module is responsible for motion segmentation. The module is responsible for maintaining the background model, updating this model and producing a 1 bit motion map. This motion map represented an a black and white IplImage represents pixels in motion as a one and pixels that are stationary as a zero.

5.2.3 Vehicle detection

This module is responsible for using the motion map to detect groups of pixels with similar motion patterns that could represent vehicles. It is responsible for thresholding the motion map (such as removing objects which are too small to be of interest) and grouping areas of similar motion into objects. This object map represented by the OpenCV data structure cvSeq is then passed to the next module.

5.2.4 Vehicle recognition

The vehicle recognition module is responsible for utilising the output of the previous modules to identify whether the objects detected in the scene are vehicles, and if they are, whether the vehicle has been seen before. In the event that a moving object is recognised as a vehicle, but not recognised as a vehicle previously seen, this module is responsible for generating a model of the vehicle that can be used for recognition in the future. This model generation can use input from a number of modules, including the video images produced by the Video Acquisition model, the motion map produced by the Motion segmentation module and the Object Map produced by the vehicle detection module. This reliance on a large number of different modules makes it the most complex module in the tracking framework. The vehicle recognition module is then responsible for passing the vehicle identifier to the vehicle tracking module along with its current position in the source video.

5.2.5 Vehicle tracking

This is the final module in the tracking framework and is responsible for maintaining information on the position of vehicles in the source video. This module processes the position information produced by the Vehicle recognition module and calculates the path of the vehicle through the source video.

5.3 Apparatus

The system must be capable of running on a desktop computer. The definition of what represents a ‘standard desktop computer’ can differ depending on the context in which the term is used. For this reason the following section outlines the exact specifications of the hardware used to complete this project.

5.3.1 Cameras

The first limiting factor of any optical tracking system is that of the video camera. Digital video cameras vary in three major areas; price, resolution and frame rate. Price refers to how much the camera costs, resolution to the level of detail of the image (pixels per frame) produced by the camera while the frame rate refers to how fast (frames per second) the camera can take pictures. Higher resolutions and frame rates generally result in higher costs. Figure 5.3 shows an overview of cameras that were evaluated for use with this project.

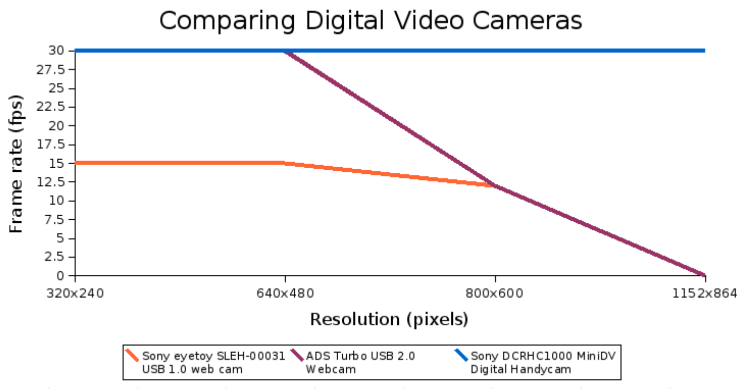

Figure 5.3: Comparing Digital video Cameras

As video resolution increases, each frame tells us more information about the real world. This increased information load means it takes longer for us to capture it and thus the frame rate decreases. The lower the frame rate of a video source, the less information about how objects are moving in a scene can be obtained. This results in the need for a delicate balance between resolution and frame rate.

5.3.2 Processor

Once a video frame reaches the application level of a tracking system, the application must process the image and track the target objects. The time required to process an image is directly related to two factors; the resolution of the image and the speed of the computer on which you are doing the processing. Figure 5.4 details the exact specifications of the computer on which all experiments have been run.

Component	**Details**
CPU	Intel Pentium IV 2.4ghz
RAM	512MB DDR333 RAM
Hard Drive	80GB HDD 7200rpm
Graphics card	NIVIDA FX 4600 128mb
Front side bus speed	533mhz
Motherboard	SUS P4B533-VM

Figure 5.4: Computer platform specifications

While the speed at which the computer can process each frame directly affects the frame rate of the tracking application, the frame rate of the application can never exceed the frame rate of the camera supplying the video. As the frame rate of the camera increases, the computer has less time available to process each frame decreases. Figure 6.1 shows the relationship between the frame rate of the camera and the time available for processing each frame.

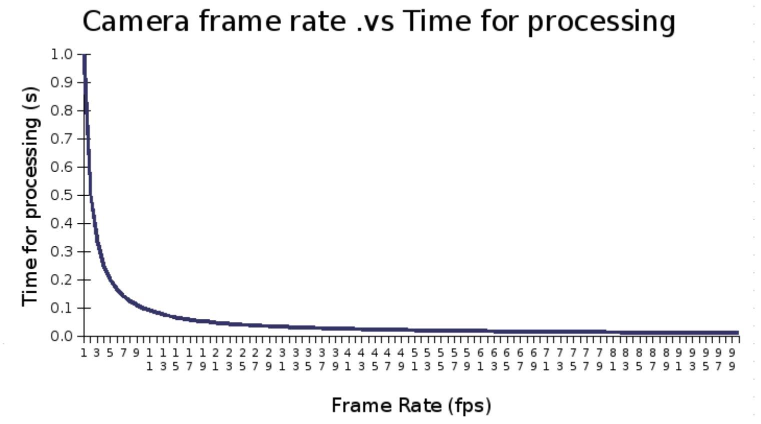

Figure 5.5: Frame rate of Camera .vs time for processing each frame

5.4 Method

The following section discusses the vehicle tracking system in detail. Results and method for each module are presented synchronously. The reason for this synchronous structure is that results of one module affect the design and results of subsequent results. Without an understanding of the output of previous modules it would be impossible to fully understand the modules that are higher up in the stack.

5.4.1 Video Acquisition

A number of different cameras have been experimented with in the development of the video acquisition module. Initially the USB 1.0 DSE XH5096 web cam was used for video capture. This camera is designed for desktop video conferencing and is the cheapest of the tested cameras. Image quality is reasonable as shown in Figure 5.6, however the camera has on-chip white balancing that can not be turned off. White balancing cameras automatically adjust the relative levels of colour in the video, so that the average colour across the image is white. Computer Vision applications in general are not compatible with white balancing cameras, as objects in the scene can rapidly alter their colours. This rapid alteration in object colour renders a large number of tracking algorithms completely useless.

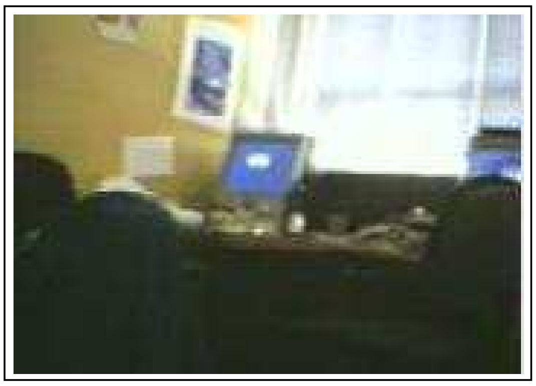

Figure 5.6: DSE XH5096 Sample output

The next low cost USB 1.0 camera tested was the Sony Eyetoy Webcam. The Eyetoy camera achieves a relatively high 640x480 frame rate by using an on-board image compression chip. This escapes the raw video band with limit of standard uncompressed video across USB 1.0 and achieves approximately 25 fps. While the camera itself works well, there is very little driver support in Windows. The camera works well after installation on Windows. However, because of the on-board JPG compression it was necessary to adjust the OpenCv library's video capture routines to include JPG decompression and conversion to IplImage formats. This significantly slowed down the frame-rate of the camera, resulting in less the 14fps when running on the desktop computer discussed in section 5.3.

The third family of cameras tested were the ADS Turbo Pro USB 2.0 and Fire-wire cameras. Initially the fire-wire card was tested. However, due to the fact that it required an additional fire-wire controller card and the orange tinge of the video as shown in figure 5.7, it was swapped for the USB 2.0 version of the camera. As shown in figure 5.7 this produced much cleaner clearer images. Both cameras had adjustable colour thresholds and it was possible to turn white-balancing off. The frame rate of both cameras was similar, with approximately 30 fps at 800x600 resolution 32 bit RGB.

While both of the ADS cameras performed well indoors, both cameras suffered from near total sun blindness when deployed outside. Sun blindness occurs when the camera cannot cope with the

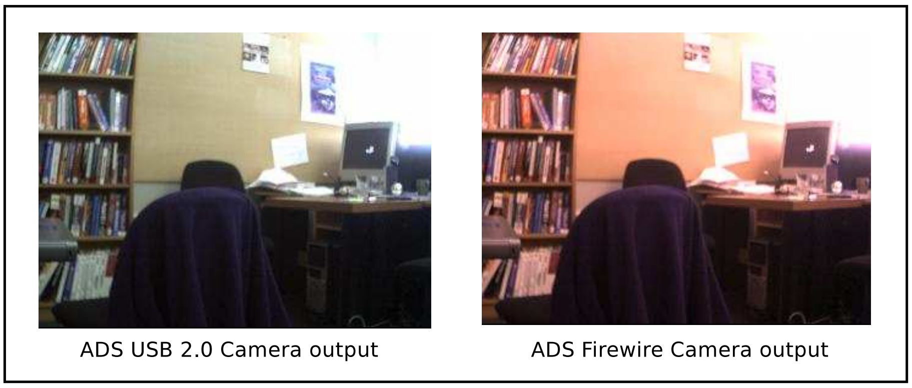

Figure 5.7: Scenes shot using the ADS Turbo Firewire and USB 2.0 cameras

increased amount of light available outside and images appear completely washed out, shown in Figure 5.8. Exposure is generally used to control the level of light allowed into the lens. However, on both the ADS Turbo models, minimum exposure was not enough to correct the problem. This means that the video is completely useless for Computer Vision applications as no objects are visible.

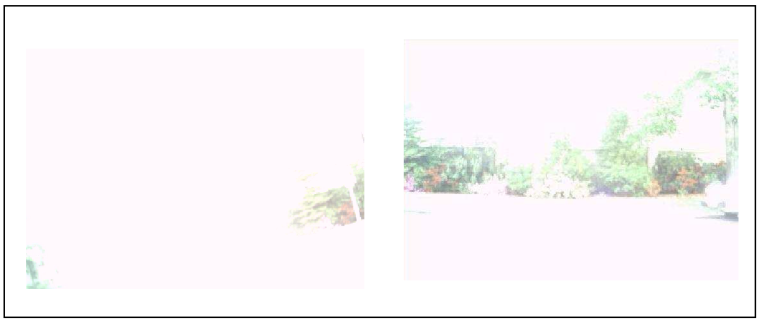

Figure 5.8: Incorrect exposure setting resulting in washed out image

The final digital video camera tested was the more expensive Sony DCRGC1000 MiniDV Digital Handycam. This camera is designed for use outdoors unlike the web cams discussed previously in this section. The automatic exposure control meant that the image was no longer washed out. In addition to the automatic exposure control white balance is configurable. The camera had a fire-wire interface so a full 50fps at 800x600 was possible. The actual configuration used during system development was 640x480 at 30fps uncompressed stream, as a higher frame rate or resolution meant subsequent modules would not operate in real time.

5.4.2 Motion Segmentation

The motion segmentation module has two distinct variants, motion segmentation using a statistical colour model and Optical flow based motion segmentation. Both methods of background subtraction are discussed in the following sections:

Statistical colour model based motion segmentation

This background subtraction method is discussed in section 4.2. After initial trials it was found that the algorithm required severe thresholding resulting in Equation 4.2.1 being modified as shown below:

$$| \left(mean_{(x,y)} - scene_{(x,y)}\right) |> threshold * \sigma_{(x,y)}$$

This new thresholded approach to background subtraction works very well indoors as shown in Figure 5.9.

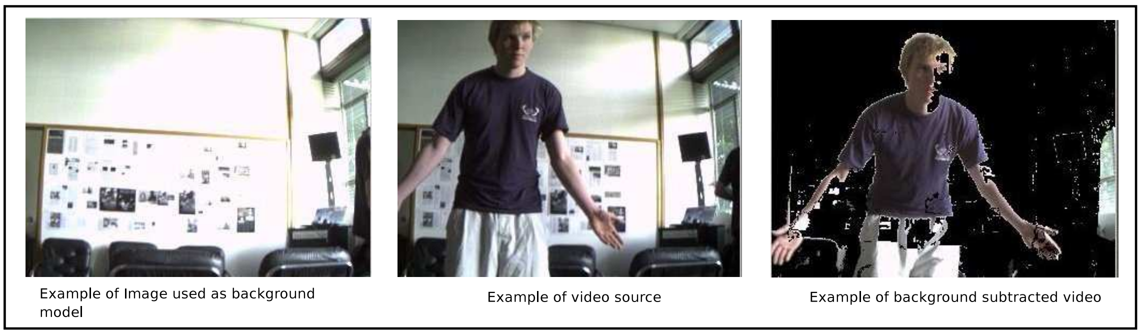
Example of Image used as background model
Example of video source
Example of background subtracted video

Figure 5.9: Simple RGB-based background subtraction indoors

Indoors light is very ambient in nature. There are many sources, and shadows are rarely visible. Outdoors there is only one predominant source of light, the sun, and as such shadows occur much more frequently. The standard background subtraction model is incapable of dealing with these shadows and the shadows are classified as being objects in motion. This problem is fully illustrated in Figure 5.10.

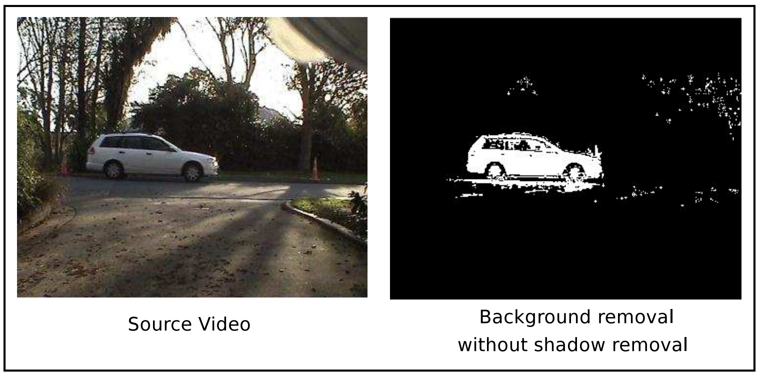

Figure 5.10: Simple RGB-based background subtraction with shadows

To solve the problem where shadows are classified as objects in motion it was necessary to add a second-pass filter to the background subtraction algorithm. This filter is based on the HSV colour model, and on the observation that shadows generally represent dull areas in the image. In terms of the HSV colour model, shadows have low intensity and saturation components. The filter can therefore classify a pixel as being motionless if both its intensity component and saturation component are below certain thresholds. The modified equation is shown below:

$$\Big(\mid \big(mean_{(x,y)} - scene_{(x,y)}\big) \mid > threshold * \sigma_{(x,y)}\Big)$$
$$\cup$$
$$\Big(Iscene_{x,y} > intesitythresh\Big)$$
$$\cup$$
$$\Big(Sscene_{x,y} > saturationthresh\Big)$$

The results are shown in the contrast between the background subtracted image before shadow removal and the image after shadow removal, both shown in Figure 5.11.

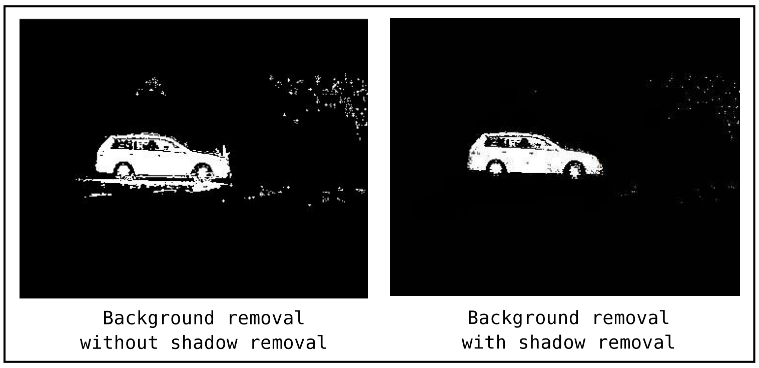

Figure 5.11: Comparing results between shadow removal and no shadow removal

Optical flow based motion segmentation

This method of background subtraction is based on the algorithms for calculating optical flow discussed in section 4.3. Background subtraction using optical flow is achieved by thresholding the velocity vector field produced by the optical flow algorithm. The resultant background subtraction algorithm defines a pixel as being stationary if its associated velocity vector has less magnitude than the threshold.

Optical flow is sensitive to changes in lighting and the algorithm is computationally expensive. The initial implementation that we developed provided much to slow to achieve any where near real time background subtraction. The OpenCV implementation which is optimised for use on Intel based machine is a lot faster, however when used outside the motion segmentation was not clean enough to be used in the final tracking application.

5.4.3 Vehicle Detection

The vehicle detection module is responsible for processing the background subtracted image. This module can be further subdivided into the following steps; object identification, object thresholding, shape approximation and shape recognition.

Object identification

This phase of the vehicle detection module processes the motion map produced by the motion segmentation module. This motion map is passed to the vehicle detection module as a black and white image. White areas of the image represent areas in motion and black areas represent, stationary or background areas. Object identification is performed using a connected components algorithm. The algorithm used is the 4-connected algorithm, which groups pixels if they are directly orthogonal.

Object thresholding

Object thresholding is the removal of groups of pixels generated by the object identification algorithm that are too small to represent the objects of interest. These small groups of pixels often represent errors in the background segmentation or movement in the background area of the video. This is a manually thresholded approach, so the user of the system must set the minimum number of pixels or size at which objects should be passed to the shape approximation algorithms. Figure 5.12, show an example of the size threshold set at different levels.

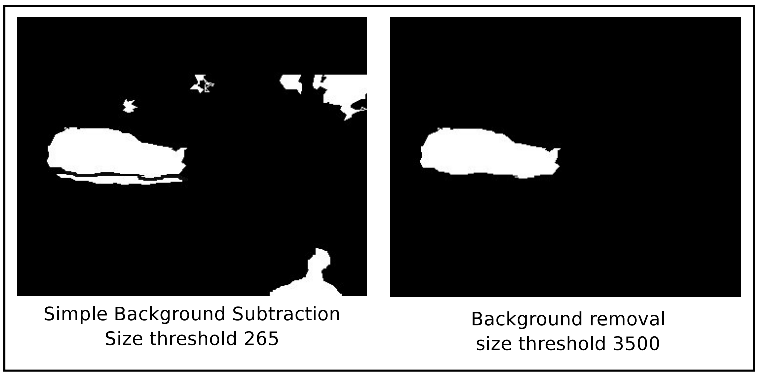

Figure 5.12: The effect of size based object thresholding on the object motion map

In addition to the removal of small objects, objects which are only partially visible are also removed. This means any contour that includes pixels at the edge of the image is removed. This is because the contour does not accurately represent the shape of the vehicle and would cause problems later in the tracking stages of the system.

Shape approximation

In order to further analyse the moving objects identified in the video stream, the next step is to generate approximations of each object's shape. Given a set of points denoted by the function $O(x, y)$ and an image $I(x, y)$ in that the object exists, it is possible to remove all points which do not lie on the border of the object. $B(x, y)$ denotes the points that belong to border of the object O, which means that B is a subset of O. The points generated by the function B can then be passed to a polygonal approximation method. These algorithms are also known as contour creation algorithms, are discussed in section 4.4. These algorithms return a further subset of the points in B that represent an approximation of the shape of the object O. During initial development the Rosenfeld-Johnston algorithm[27] was used to produce the vehicle contours. This algorithm requires manual thresholding of the parameter m (see Section 4.4 for more information), and often produces excessively rough shape estimations. The variations in polygonal approximations generated by this algorithm are shown in Figure 5.13. The Teh Chin polygonal approximation algorithm[5] was then tested. This algorithm does not require the manual thresholding and as such produces a much more reliable shape approximation. The results of this polygonal approximation algorithm are also shown in Figure 5.13.

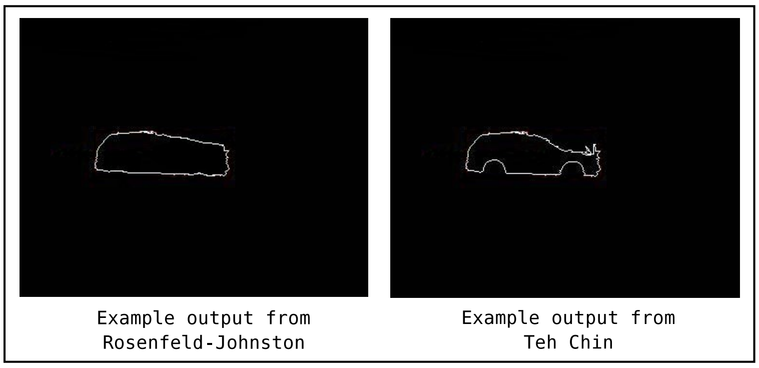
Example output from Rosenfeld-Johnston
Example output from Teh Chin

Figure 5.13: Comparison of polygonal approximation methods[27][5]

5.4.4 Vehicle Recognition

Once the contours representing the objects in the source video have been created, the next step is to evaluate the optical properties of the object. This evaluation is accomplished in two phases; The initial shape based thresholding and the secondary vehicle recognition based on shape and colour.

Shape based thresholding

For each vehicle the contour is draw as a binary silhouette. The set of geometric moments (see section 4.5) are then calculated. These geometric moments are then used to obtain information about the object.

- m_{00}, the order zero geometric moment, is used to calculate the area of the object in the source video
- m_{01} and m_{10}, the first order moments, are used to calculate the centroid or optical center of mass of the object in the video frame.

These geometric moments are then used to calculate the central geometric moments of the object (as described in Section 4.5). The centralised geometric moments are then used to generate a set of seven invariant feature descriptors that can be used for shape similarity evaluation. A library of three very simple contours is maintained in memory and the object's contour is compared to these to evaluate the likely vehicle type. This set of basic vehicle types is shown in figure 5.14.

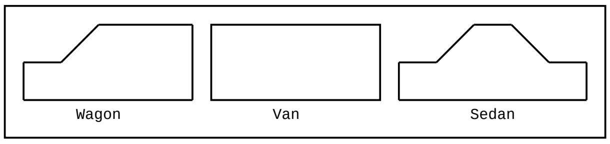

Figure 5.14: Simple vehicle types

This produces a match metric which varies between zero and one. Each of the vehicle models is compared to the vehicle contour and the highest value is determined to be the most likely format of the vehicle being tracked. Once again, a threshold is applied. If the match metric of all vehicle types is less than the threshold value, the object is discarded. The reason for this thresholding is two-fold; Initially this removed large moving objects that do not represent vehicles from the video, for example trees swaying in the background or people walking on the pavement.

Secondly this limits the number of comparisons needed to correctly identify a vehicle, by assigning a vehicle a class. This is best explained with an example, If the system begins to track a vehicle and this vehicle is recognised as being of shape similar to a van, then there is no need to check the database to see if the vehicle matches any of the vehicles in the sedan or wagon classes. This instantly reduces the number of comparisons needed and thus increases the efficiency of the tracking algorithm

Shape and Colour based vehicle recognition

After the initial shape-based thresholding is complete, the secondary recognition phase begins. This stage introduces the use of colour-based tracking. Without this colour based approach it would be possible for two similar model vehicles to pass the tracking system and be classified as the same vehicle, irrespective of the fact that one car was green and the other red. Colour-based tracking is achieved by using the pixels within the contour representing the vehicle being tracked to generate a RGB colour histogram. The vehicles contour is then compared to the contour of each vehicle previously seen in the video stream. If the contour does not matches within a given threshold *(manually set)* then a new vehicle entry is created in the database. This vehicle entry includes the vehicles contour and the pre-calculated geometric moments, centralised moments, seven invariant feature descriptors and the colour histogram. If the vehicle's contour does match with a contour in the vehicle database then the contour image of the vehicle is back projected using the colour histogram of the vehicle in the database with which it matched with *as detailed in section 4.6*. If the number of pixels that are back projected in the vehicle's contour is higher than a manually set threshold then the match is deemed to be correct. The vehicle in the database, then has its path updated with the new position of the vehicle's center of mass.

5.4.5 Vehicle Tracking

This module is primarily responsible for rendering the path of a vehicle on a screen. When a vehicle enters the video stream, a new entry is created in the database by the vehicle recognition module. The position of the center of mass is then rendered by the tracking module on an image. As the vehicle moves through the video frames the path is updated and rendered to the screen. This module is also responsible for removing vehicle identifiers that have been stagnant for a set period of time. Examples of vehicle paths generated are shown in Figure 5.15.

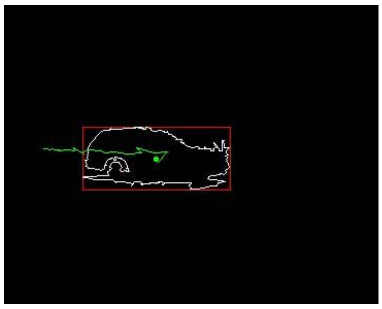

Figure 5.15: Example vehicle path

Initially the period of time that a vehicle remains in the database was set by a manual threshold, however a better approach was developed. By generating an orange biased colour histogram artificially we can locate two road cones placed 20 meters apart on the road using back projection see figure 5.16.

This enables to calculate how many pixels represent 20 meters in the real scene. As mentioned previously, as the vehicle moves through the scene the position of the vehicle's center of mass is stored. We can then calculate the relative velocity of vehicle in the scene. This velocity can then

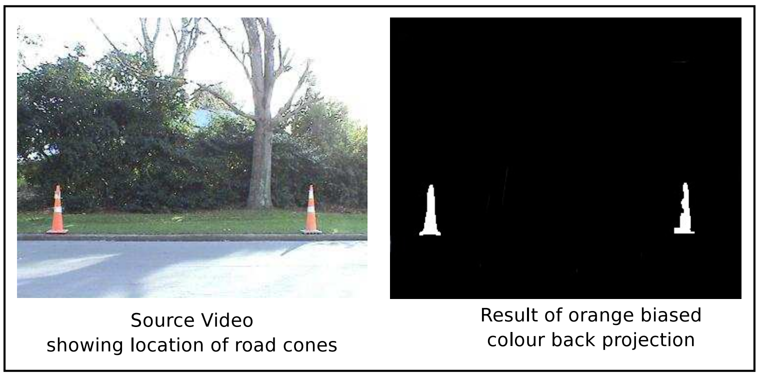

Figure 5.16: Example of road cone back projection

be used to predict the length of the time the vehicle will remain in the video stream. When this time decreases below a certain threshold, the vehicle is removed from the database.

Chapter 6

Summary of Results

This section gives an overview of the results obtained through the development of this research. Because of the young state of Computer Vision as a field of Computer Science there is very little in the way of formalised methods for evaluating tracking solutions. Due to the modular nature of the system, a cyclic development model was used. This means both that each module was developed in order initally and that results from one model affected the design of subsequent modules. Chapter 5 System Development presents the results of each module individually simultaneously with an explanation of their design. This chapter summarises these results and looks at the system as a whole. This section presents information obtained by manual analysis of a 30 second clip of sample video. This sample video begins with no objects visible, after 5 seconds have elapsed a single white wagon enters the video stream travelling at approximately 15 km/hr. It takes approximately 7 seconds to pass through the video stream at which point there are no vehicles visible. At 15 seconds a second vehicle enters the video stream and exits at approximately 22 seconds. For the last 8 seconds of the video there are no visible vehicles. This section is divided into two section; performance related results and quality related results.

6.1 Performance Results

The primary measure of performance for a tracking application is frame rate of the tracking solution. Figure 6.1 shows the frame rate of the tracking solution for the 30 second sample solution. If the frame rate drops below that of the camera, then video frames are simply discarded until the tracking modules are ready to accept the next frame. By looking at this graph it becomes

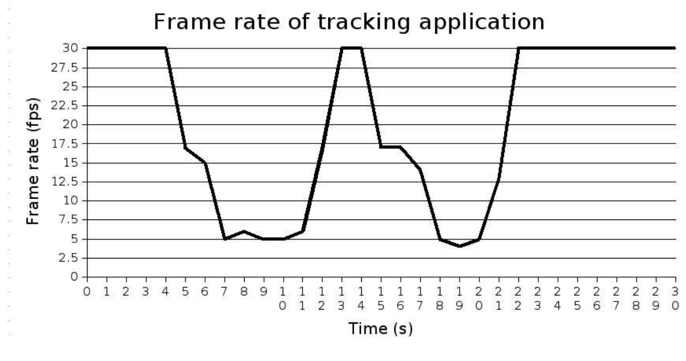

Figure 6.1: Frame rate of tracking application across sample video

obvious that as the initial vehicle enters the video stream the frame rate dramatically decreases. Once again as the second vehicle enters the video stream the frame rate decreases again. This is due to the fact that when there is no movement in the video stream then the layers above the Background subtraction model do not operate. Once an object does enter the video stream then a significant amount of processing time is required to correctly track the vehicle.

6.2 Quality related Results

There are two quality related results that can be easily calculated from the 30 second example video clip. The accuracy of our of background subtraction and the accuracy of our velocity calculations.

It is possible to compare the background subtraction algorithms by manually calculating the number of pixels that make up the vehicle in the source video and then comparing them to the number of pixels contained within the vehicles contour. The resulting statistics let us analyse the quality of our background subtraction model, thereby also analysing those modules build on top of the background subtraction module.

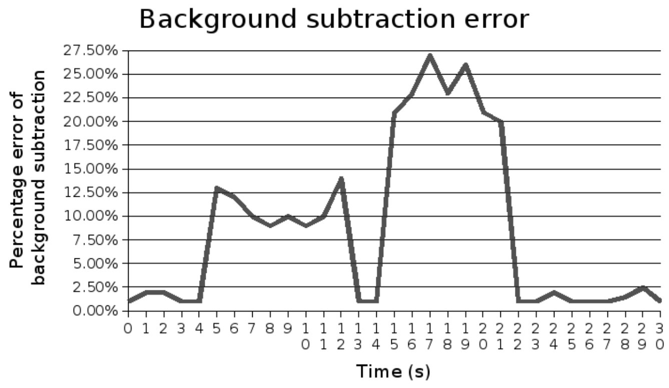

Figure 6.2: Comparing background subtraction methodologies

Finally, the last module in the tracking application is the Vehicle tracking module. This module is responsible for making real world inferences about vehicle movement based on the information obtained from those modules beneath it. The following graph demonstrates the accuracy of the vehicle velocity calculations.

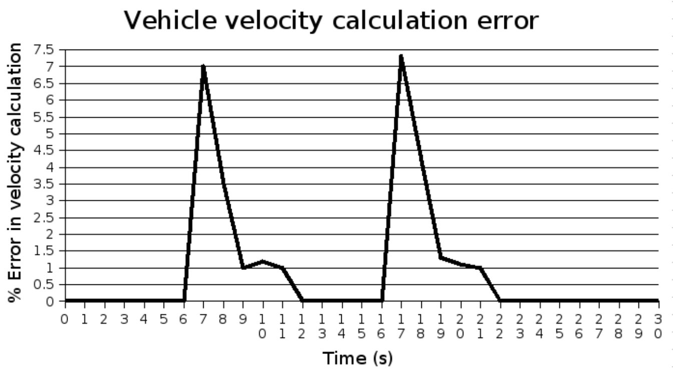

Figure 6.3: Velocity calculation error

Chapter 7

Conclusion

The result of this research is a modularised approach to object tracking. This enables easy comparison of algorithms and fast prototyping of computer vision applications. We have shown that at each stage in the development process a number of options can be rapidly prototyped and tested. In combination with the openCV SDK which provides a large number of computer vision algorithms and a reliable framework for video capture, we have produced a simple, yet reliable solution to vehicle tracking. We have improved the default background segmentation algorithm provided by openCV by implementing a second pass shadow removal algorithm. This shadow removal algorithm is based on the simple observation that shadows have the same hue with a lower intensity than the surrounding environment. For this reason the algorithm is based on the HSV colour model in which the second and third components correspond directly to the saturation and intensity of the colour. Pixels are regarded to be in shadow if they fit inside the bounds of the initial RGB background model but have a low intensity and saturation components when represented in the HSV colour model. Due to the fact that the position of our camera is orthogonal to the road plane after background segmentation, we achieve accurate two dimensional silhouettes. This means that the process of vehicle recognition is more robust on our system than it is with other systems where the camera is located in an elevated position. In addition to this efficient platform design our vehicle tracking and recognition phases are dependant on two distinctly different algorithms; Shape based recognition (reliant on Teh Chin polygonal approximation and Hu invariants) and Colour based recognition (using colour histograms and back projection). Both of these approaches have been utilised singularly for object tracking and reinforce each other providing the system with a more robust overall tracking approach.

There are two primary aspects to the research discussed in this report. The development of a tracking framework and the development of a system built on top of this framework. The design of the framework, the modular approach to vehicle tracking has few limitations. The tracking system however has a much larger set of limitations. When evaluating the system as a whole there a number of limitations. Initially when the system is deployed it is necessary to capture a significant amount of un-occluded background video, as the complexity of the background scene changes the higher the number of frames needed. This is because the number of frames captured corresponds directly to the quality of the background colour model used for background subtraction. Secondarily, the system does not cope very well with occlusion. If two cars enter the scene at different ends of the video stream then the system is capable of differentiating between them. The problem occurs however if the cars are always slightly occluded, the system is unable to fit a vehicle model to the single optical feature representing two vehicles and thus discards the object. In addition to this weakness, tracking calculations such as vehicle velocity are based on the frame rate of the camera, should the frame rate of the camera change at any point, then all velocity calculations would become inaccurate. Each of these errors while small by themselves, when combined mean that further development would be needed before any commercial deployment could be considered.

The modular design of our system means that it would be possible to extend it in the future with other algorithms at any stage in the pipeline. Each of these modules could be improved in a number of different ways, a subsection of the more important improvements are presented below:

- Video Acquisition
  - Integration of stereo camera depth information to improve occulsion handling
- Motion Segmentation
  - Adaptation of the currently used background removal algorithm currently used to include the ability to reclassify objects as being part of the background if they maintain a static location in the scene for a given period of time.
- Vehicle Detection
  - Automation of minimum and maximum object size thresholds. Possibly by calculating average size of vehicles being tracked and removing all objects outside of 2 standard deviations from this mean.
- Vehicle Recognition
  - Use of more accurate two dimensional models for inital vehicle classification
  - Evaluation of Earth Mover Distance comparison metric for coloured histogram matching.
- Vehicle Tracking
  - Intergration of Particle filter to vehicle tracking.

Vehicle tracking solutions have a large range of possible applications. Perhaps the most important applications are security related. Given a system such as the one presented in this paper, it is possible to monitor the speed and position of a number of vehicles in a given area. This can be used to alert security personnel to suspicious vehicles, such as those parked too close to vital buildings or left stationary for a given period of time. This system could also be used to locate all vehicles of a certain colour or shape in any number of separate video streams, helping with the location of stolen vehicles. In addition to these security applications, a larger set of more mundane applications also exist. One example is an automated parking monitor for a car park. If a system, such as the one presented in this paper was placed at all the entrances and exits to a complex such as Canterbury University, a tally of the total number of cars currently inside the grounds could be collected. If this number exceeded the total number of car parks available, signs at the entrances could indicate that it was unlikely that a park would be available for any new vehicles entering the complex.

Bibliography

[1] Forsyth an d Ponce. *Computer Vision - A Modern Approach*. Prentice Hall, 2003.

[2] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters, 1997.

[3] Lisa M. Brown. View independent vehicle/person classification. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 114–123. ACM Press, 2004.

[4] T. Camus. Real time quantized optical flow. *The Journal of Real-time Imaging (Special issue on Real-Time motion analysis)*, 1997.

[5] R.C. C.H. Teh. Detection of dominant points on digital curves. *IEEE*, 1989.

[6] J. et al Connel. Detecting and tracking in the ibm people vision system. *ICME*, 2003.

[7] Smith F.W and Wright M.H. Automatic ship photo interpretation by the method of moments. *IEE Trans. on Computers*, 20(9):1089–1095, 1971.

[8] A D Worall C I Attwood GD Sullivan, K D Baker and P R REmagnino. Model-based vehicle detection and classification using orthographic approximations. *IEEE*, 1996.

[9] Markus Beier Gerald Kuhne, Stephan Richter. Motion-based segmentation and contour-based calssification of video objects. *Praktische Informatik IV Universtity of Mannheim L 14,16, 68131 Mannheim, Germany*, 2001.

[10] Young-Kee Jung; Yo-Sung Ho. Traffic parameter extraction using video-based vehicle tracking. *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, 1999.

[11] B. G. Horn, B. K. P. & Schunck. Determining optical flow. *AI Memo 572, Massachusetts Institute of Technology*, 1980.

[12] B.G. Horn, B.K.P; Schunck. Determining the optical flow. *Artificial Intelligence*, 17:185–204, 1980.

[13] Jianguang Lou; Tieniu Tan; Weiming Hu. Visual vehicle tracking algorithm. *Electronics Letters*, 38(18):1024–1025, 2002.

[14] M. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 1962.

[15] Luc Van Gool Katja Nummiaro, Esther Koller-Meier. Object tracking with an adaptive color-based particle filter. *Katholieke Universiteit Leuven, ESAT/VISICS, Belgium*, 2001.

[16] Tomas Svoboda Katja Nummiaro, Esther Koller-Miere. Color-based object tracking in multi-camera environments. *Katholieke Universiteit Leuven, ESAT/VISICS, Belgium*, 2003.

[17] C.W.; Lee K.M.; Yun T.S.; Kim H.J. Kim, J.B.; Lee. Wavelet-based vehicle tracking for automatic traffic surveillance. *Electrical and Electronic Technology, TENCON. Proceedings of IEEE Region 10 International Conference on*, 1(19-22):313–316, 2001.

[18] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *ECCV (1)*, pages 189–196, 1994.

[19] H.-H. Leuck, H.; Nagel. Model-based initialisation of vehicle tracking: dependency on illumination. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, 2001.

[20] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video, 1998.

[21] John MacCormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, 2000.

[22] M. Nagel, H.-H.; Haag. Bias-corrected optical flow estimation for road vehicle tracking. *Computer Vision, 1998. Sixth International Conference on*, 1998.

[23] M.; von Seelen W. Noll, D.; Werner. Real-time vehicle tracking and classification. *Intelligent Vehicles '95 Symposium., Proceedings of the*, pages 101–106, 1995.

[24] S.; Kosuge Y. Okada, T.; Tsujimichi. Multisensor vehicle tracking method for intelligent highway system. *SICE 2000. Proceedings of the 39th SICE Annual Conference. International Session Papers*, pages 291–296, July 2000.

[25] K R Ramakrishnan R Mukundan. *Moment Functions in image Analysis - Theory and Applications.* World Scientific, 1998.

[26] Zhimin Fan; Jie Zhou; Dashan Gao; Gang Rong. Robust contour extraction for moving vehicle tracking. *mage Processing. 2002. Proceedings. 2002 International Conference on*, 3(24-28):625–628, 2002.

[27] A. Rosenfel and E. Johnston. Angle detection on digital curves. *IEE Trans Computers*, 22:875–878, 1973.

[28] Russel and Norvig. *AU, A Modern approach.* Prentice Hall, 1995. Figure 24.9, pg. 737.

[29] Dudani S.A. Aircraft identification by moment invariants. *IEEE Trans. on Computers*, 26(1):39–45, 1977.

[30] M Schmid. An approach to model-based 3-d recognition of vehicles in real time by machine vision. *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, 1994.

[31] Steve Sengels. Explanation of lagrangian multipliers. `http://www.cs.toronto.edu/~sengels/tutorials/lagrangian.html`.

[32] S.M. Smith. Asset2:real-time motion segmentation and shape tracking. *Proc Int. Conf. on COmputer Vision 4*, 1995.

[33] Ching-Po Lin; Jen-Chao Tai; Kai-Tai Song. Simultaneous tracking of pedestrians and vehicles by the spatio-temporal markov random field model. *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003.

[34] Ching-Po Lin; Jen-Chao Tai; Kai-Tai Song. Traffic monitoring based on real-time image tracking. *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 2003.

[35] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[36] Weiming Hu; Xuejuan Xiao; Dan Xie; Tieniu Tan. Traffic accident prediction using vehicle tracking and trajectory analysis. *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE , Volume: 1 , 12-15 Oct.*, 2003.

[37] M.B.; Groen F.C.A. van Leuven, J.; van Leeuwen. Real-time vehicle tracking in image sequences. *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE, Vol.3, Iss., 2001.*

[38] S. Weiming Hu; Xuejuan Xiao; Xie, D.; Tieniu Tan; Maybank. Traffic accident prediction using 3-d model-based vehicle tracking. *Vehicular Technology, IEEE Transactions on*, 2004.

[39] L.J. Guibas Y. Rubner. C. Tomasi. The earth mover s distance as a metric for image retrieval. *Technical Report STAN-CS-TN-98-86*, 1998.